IMT Institute for Advanced Studies, Lucca

Lucca, Italy

Extending Ontology Queries with Bayesian Network Reasoning

PhD Program in Computer Science and Engineering

XX Cycle

By

Bellandi Andrea

2008

The dissertation of Bellandi Andrea is approved.

Program Coordinator: Prof. Ugo Montanari, University of Pisa

Supervisor: Prof. Franco Turini, Department of Computer Science - University of Pisa

Supervisor: ,

Tutor: ,

The dissertation of Bellandi Andrea has been reviewed by:

Prof. Yun Peng, Department of Computer Science and Electrical Engineering - University of Maryland

Thierry Declerck, DFKI GmbH Language Technology Lab - Germany

IMT Institute for Advanced Studies, Lucca

2008

To my uncle Renzo Orsini

Contents

List of Figures

List of Tables

March 26, 1976	Born, Livorno, Italy
October 2004	Degree in Computer Science Final mark: 110/110 University of Pisa, Italy

Publications

1. A. Bellandi, B. Furletti, V. Grossi, A. Romei. “Ontology-driven association rules extraction: a case study,” In the proceedings of the *Workshop Context and Ontologies: Representation and Reasoning*, 2007.
2. A. Bellandi, B. Furletti, V. Grossi, A. Romei. “Pushing Constraints in Association Rule Mining: An Ontology-Based Approach,” In the proceedings of the *IADIS International Conference WWW/INTERNET*, 2007.
3. M. Baglioni, A. Bellandi, B. Furletti, L. Spinsanti, F. Turini. “Ontology-Based Business Plan Classification,” In the proceedings of the *12th IEEE International EDOC Conference. The Enterprise Computing Conference (EDOC 2008)*, 2008.
4. A. Bellandi, B. Furletti, V. Grossi, A. Romei. “Ontological Support for Association Rule Mining,” In the Proceedings of the *IASTED International Conference on Artificial Intelligence and Applications (AIA 2008)*, 2008.
5. M. Baglioni, A. Bellandi, B. Furletti, C. Pratesi, F. Turini. “Improving the Business Plan Evaluation Process: the Role of Intangibles,” To be published in the I*nternational Journal Quality Technology and Quantitative Management - Special Issue on Non-Standard Analysis of Customer Satisfaction Survey Data*, 2008.
6. D. Bacciu, A. Bellandi, B. Furletti, V. Grossi and A. Romei “Discovering Strategic Behaviors in Multi-Agent Scenarios by Ontology-Driven Mining,” Chapter of the book **Robotics, Automation and Control**, ITECH Books http://www.i-techonline.com/ - To be published, 2008.

Abstract

Today, ontologies are the standard for representing knowledge about concepts and relations among concepts concerning specific domains. In general, ontology languages are based on crisp logic and thus they can not handle incomplete or partial knowledge about application domain. However, uncertainty exists in domain modeling, ontology reasoning, and concept mapping. Our choice for dealing with uncertainty, is the Bayesian probability theory. The technique of representing an ontology by means of a bayesian network is used for having a new knowledge base enriched with uncertainty, over which making inference and probabilistic reasoning. Our method is composed of three steps. The first one is to compile the ontology into a bayesian network. We define the ontology compiling process for extracting the bayesian network structure directly from the schema of the knowledge base. The second one is to learn the initial probability distributions. We provide a computation process for learning the probability distributions, both prior and conditional, directly from the ontology instances, based on the Bayes theorem. The third one is to provide a bayesian query language for answering queries involving probabilities. Although evaluating bayesian networks is, in general, NP-hard, there is a class of networks that can efficiently be solved in time linear in the number of nodes. It is that of the polytree. On the basis of this bayesian network class, it is provided a bayesian query language for answering queries involving probabilities concerning both $is - a$ ontology relations and $object - property$ relations. It is based on recursive algorithms implementing top-down and bottom-up reasoning over polytrees networks.

Chapter 1

Introduction

Today, ontologies are the standard for representing knowledge about concepts and relations among them in specific domains. The term ontology came from Philosophy and was applied to Information Systems in the late seventies to characterize the formalization of a body of knowledge describing a given domain. One of the major drivers for the popularity of that concept was the realization that many of the most challenging problems in the information technology field (e.g. interoperability among systems) cannot not be solved without considering the semantics intrinsically embedded in each particular knowledge system. In other words, addressing issues such as $type\ consistency$ or $consistency\ of\ format$ might sometimes work in very controlled environments, but successful knowledge sharing requires $semantic\ interoperability$, a much stronger requirement.

Commonly touted as a promising solution to the problem of semantic interoperability (e.g., (Cha99)), ontologies are formal representations of knowledge about a given domain, typically expressed in a manner that can be processed by machines. Specifically, an ontology explicitly represents the types of entities that can exist in the domain, the properties these entities can have, the relationships they can have to one another, the roles they can play with respect to one another, how they are decomposed into parts, and the events and processes in which entities can participate. Ontologies are useful for ensuring that producer and consumer share a common interpretation of data, especially in situations in which ownership boundaries are crossed.

A number of ontology definition languages has been developed over the past few years. The emerging standard one, recommended by W3C, is the Ontology Web Language (OWL). OWL has been designed to meet this need for a Web Ontology Language and it is part of the growing stack of W3C recommendations related to the Semantic Web.

- XML provides a surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.
- XMLSchema is a language for restricting the structure of XML documents and also extends XML with datatypes.
- RDF is a datamodel for objects (*resources*) and relations between them, it provides a simple semantics for this datamodel, and these datamodels can be represented in an XML syntax.
- RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes.
- OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. *exactly one*), equality, richer typing of properties, characteristics of properties (e.g. *symmetry*), and enumerated classes.

In general, ontology languages are based on crisp logic and thus they can not handle incomplete or partial knowledge about application domains. However, uncertainty exists in almost every aspect of ontology engineering, for example in domain modeling, and ontology reasoning. Not surprisingly, as ontology engineering research has achieved a greater level of maturity, the need for representing uncertainty in ontologies in a principled way has became clear. There is increasing interest in extending traditional ontology formalisms to include sound mechanisms for representing and reasoning with uncertainty (CK05), (Las07). Initial attempts to represent uncertainty in ontology languages tend to begin with constructs for attaching probabilities as attributes of entities. This approach is clearly inadequate, in that it fails to account for structural features such as conditional dependence (or independence), double counting of influence on multiply connected graphs, and context-specific independence. Many researchers have pointed out the importance of structural information in probabilistic models (e.g. (Sha86), (Pea88), (Sch94), (Kad96)). In short, annotating an ontology with numerical probabilities is just not enough, as too much information is lost to the lack of a representational scheme that can captures structural nuances of the probabilistic information.

An important question, therefore, is how probabilistic formalisms such as Bayesian Networks can be merged with formal ontologies. Probabilistic theories produce qualified conclusions, graded by numerical measures of plausibility. By contrast, formal ontologies have focused on purely logical reasoning that leads to definite conclusions. Formal ontological categories are related to one another in definite, law-governed ways, and are understood as possessing a purely binary truth functionality. Formal ontologies are useful for data integration, particularly at the upper-most levels, because they provide for a logical structure for various categories and relations, independent of any particular material knowledge of a given domain (Smi95). In this sense, a formal ontology provides a means of understanding all types of objects, attributes and relations associated to one another within a given domain by understanding the most basic formal structures they share in common (Lit06a). The question of how formal ontologies can be merged with probabilistic reasoning rests on first defining which items are in an ontology per se and which items are associated with the ontology (e.g., the reasoning engine, the query language, the results analyzer, etc.). An upper ontology provides asserted facts about the ontic world, meaning the world of general being as opposed to any distinctive philosophical or scientific theory of that world, i.e., the ontological. So, according to this approach, an upper ontology provides a type of assumed god's eye view of reality, independent of human observations. By their very nature, human observations presume certain epistemic (i.e., mind- or knowledge-dependent assertions about reality (e.g., as discussed in the lengthy philosophical debates between realist and conceptualist theories of reality) (Lit06b), (Lit05). At the upper-most levels, for example, an ontology normally contains non-recursive categorical relations such as: a TerroristAgent $is-a$ Person, an IED $is-a$ Explosive, an ObjectShape is dependent on Substance). The lattice of types and subtypes is a logical structure that is generally taken as given by both logical and probabilistic domain theories. While there may be competing upper ontologies, each with its own type lattice, generally within an ontology there is no uncertainty associated with the categorical relations. However, the situation changes when we consider the problem of categorizing instance data. As an example, consider an individual who is declared a $Person$-Of-$Interest$, and is being observed to assess whether or not he is engaging in terrorist activities. Information relevant to this problem includes, for example, the network of individuals with whom he associates, his religious affiliation, purchases he has recently made (e.g., materials that could be used to manufacture explosive devices), phone calls to individuals suspected of plotting an attack, etc. The decomposition of Person-Of-Interest into sub-categories of Terrorist and non-Terrorist is a purely logical assumption. However, categorizing an individual as a terrorist or non-terrorist would make use of probabilistic information, such as the base rate of terrorist versus nonterrorist individuals within the relevant population, the likelihood of the pattern of attributes and activities given that the individual is a terrorist versus a non-terrorist, and the credibilities of the reports on which we are basing our inferences about his attributes and activities (e.g., (Wri06)). Probability is an essential tool for performing this kind of inference in a systematic and principled manner. To perform this kind of reasoning, a system needs the basic categorical knowledge typically encoded in an ontology, and also the likelihood information needed by the probabilistic reasoner. This likelihood information can be obtained from statistical summaries of past instance data, from the judgment of experienced experts, from physical characteristics of sensing systems, or from some combination of the above. This information must be represented in computational form to be processed by probabilistic reasoning algorithms. Increasingly, with the proliferation of distributed fusion systems and web services, it is becoming important to represent this likelihood information not just internally within a given system, but also for consumption by other systems with which it interoperates. Data quality information must be represented as metadata associated with a web service. When a service returns a result on a situation-specific query, it often must return not just a most likely conclusion, but also information on the uncertainty associated with the conclusion, and also pedigree information to provide the consumer with an audit trail regarding how the conclusion was reached. Interoperating systems require not just shared vocabulary for domain concepts, but also shared vocabulary for communicating statistical regularities pertaining to categories in the ontology, as well as uncertainties associated with instance-level reasoning results, and pedigree information about how conclusions were reached.

Our reasearch aims to extend ontology querying for handling uncertainty, without extending the ontology representation formalism. The uncertainty is derived directly from the knowledge base (which is precise knowledge), and no modification and no intial probability distributions for answering to queries involving probabilities are required. This research develops an extension of the ontology queries with Bayesian network reasoning. Our approach does not need to modify or re-arrange the original knowledge base in terms of both schema and instances for dealing with uncertainty, by introducing new relations, or using non-classical Bayesian networks. In the following we briefly discuss our approach.

1.1 The approach

This research aims to define a Bayesian Query Language (BQL) which extends ontology query languages. So, our natural choice for dealing with uncertainty, is the probability theory, in particular the bayesian one. We are attracted to bayesian networks in this work also for the structural similarity between the DAG of a bayesian network and the RDF graph of an ontology: both of them are directed graphs, and direct correspondence exists between many nodes and arcs in the two graphs. In general, many real-life situations can be modeled as a domain of entities represented as random variables in a probabilistic network. A probabilistic network is a clever graphical representation of dependence and independence relations among random variables. Supposing, for example, that our knowledge is about businesses, we could want to be able to answer queries like the following ones:

- which is the likelihood of default of a company given that it is limited and has branches outside Europe?
- which is the likelihood of a particular project type given that it is led by male managers working for a ltd company?

The whole proposed method is composed of three steps:

- Compiling the ontology into a bayesian network. We define the ontology compiling process for extracting the bayesian network structure directly from the schema of the knowledge base.
- Learning the initial probability distributions. We provide a computation process for learning the probability distributions, both prior and conditional, directly from the ontology instances, based on the Bayes theorem.
- Providing a Bayesian query language for answering queries involving probabilities. In these terms, an advantage of our approach is that, compiling an ontology into a classical bayesian network, it is possible to provide a query language based on well-know inference schemas over bayesian networks.

In our view an ontology is a set of taxonomies and relationships among them. According to RDF/OWL definitions, we can distinguish two kinds of relations: the first one is the $is - a$ relation, binding classes within each taxonomy, and the second one is the $object\ property$ relation, binding a class to another one. Our ontology compiling process produces a two-levels bayesian network. Note that, in general, but depending on the specific ontology schema describing the business domain, example queries could involve both $is - a$ relationships (e.g., limited company $is - a$ company, and male $is - a$ person) and ontology relationships among concept classes (e.g., having branches, leading projects). In our context, we treat the probability as the chance that an ontology relationship, either $is-a$ or $objectproperty$, exists among ontology instances. Then, on the basis of the ontology instances, the network learns the probability distributions by using the Bayes theorem. From this point of view, we extract the uncertain knowledge directly from the original knowledge base, and just implicitly coded in it. So, it is not necessary to modify the information representation, or adding new information for introducing the uncertainty, and extracting the related bayesian network. In general, a bayesian network of n variables consists of a DAG (Direct Acyclic Graph) of n nodes and a number of arcs. Nodes t_i in a DAG correspond to random variables, and directed arcs between two nodes represent direct causal or influential relations from one variable to the other. Our process codes each taxonomy k, into a bayesian network B_k, at the first level. At the second level the resulting bayesian network has all the B_k as nodes and all the relations among taxonomies as annotated arcs. For this reason, we introduce a two-levels Bayesian network ($2lBN$), and its structural part codes the two levels specified so far. The uncertainty of the causal relationships is represented locally by the conditional probability tables (CPTs) in terms of both instances belonging to an ontology class, referring to each B_k network, and triples belonging to ontology statements, referring to relations among taxonomies. All of these CPTs are coded in the $2lBN$ probabilistic part. In general $P(t_i|father(t_i))$ is associated with each node t_i, where $father(t_i)$ is the parent set of t_i. Under the conditional independence assumption, the joint probability distribution of $t = (t_1, t_2, ..., t_n)$ can be factored out as a product of the CPTs in the network, namely, the chain rule of BN:

$$P(t) = \sum_i P(t_i|father(t_i)).$$

Although evaluating bayesian networks is, in general, NP-hard, there is a class of networks that can be efficiently solved in time linear in the number of nodes. The class is that of polytrees. It is one in which the underlying undirected graph has no more than one path between any two nodes (the underlying undirected graph is the graph that one gets if one simply ignores the directions on the edges). In particular, for each node t, it is possible to partition all the other nodes into two disjoint sets. The first one, is the set of nodes which are over t, that is those nodes that are connected to t only via the $fathers$ of t. The second one is the set of nodes which are under t, that is those nodes that are connected to t only by the immediate descendents of t. On the basis of this bayesian network class, it is provided a BQL for answering queries involving probabilities concerning both $is - a$ ontology relations and $object\ property$ relations. BQL is based on recursive algorithms implementing top-down and bottom-up reasoning over polytrees networks.

1.2 Thesis Outline

This thesis is organized as follows. In chapter 2, section 2.1 provides an introduction to what an ontology is, and how it is used for representing knowledge; section 2.2 introduces some basics about Bayesian networks, by discussing the fundamental concepts of the graphical language used to construct probabilistic networks, the uncertainty calculus used in probabilistic networks to represent the numerical counterpart of the graphical structure. We introduce the notion of discrete Bayesian networks and the principles underlying inference in probabilistic networks; section 2.3 discusses a comparison of our work with the literature. Chapter 3 introduces the two-levels Bayesian network compiling process; section 3.1 gives an overview about the compiling process; section 3.2 introduces the definition of the two-levels Bayesian network, and section 3.3 specifies the ontology requirements treated by our method; section 3.4 defines the compiling process in detail. Chapter 4 introduces the reasoning schemas for making inference on the two-levels Bayesian networks, showing some examples of query computation. In particular, section 4.1 defines the Bayesian query language in terms of its syntax and its operational semantics. Other sections show how make inference over taxonomies, taxonomies with multiple inheritance, and polytrees. Chapter 5 presents the conclusion of this research and discuss open problems and future work.

Chapter 2

Background and Related Works

2.1 Ontology

In the next sections we provide a general introduction to ontology, its formal definition, and the ontology standard language of the semantic web.

2.1.1 Introduction

The ontology may be defined as the study of being as such. The introduction of the word itself has remote origins and may be seen as an attempt to modernize the classical debate about metaphysics. One of the first exponent of this new discipline was Aristotle that, in (ABC), presents a first work that can be resembled to an ontology. He provides a list of categories that represent an inventory of what there is, in terms of the most general kinds of entities. Later on, the Latin term ontologia appeared for the first time in the circles of German protestants around the 1600 (LJTP), while the first English appearance goes back to 1721 in the Baileys dictionary in which it was defined as an Account of being in the Abstract. The first use of the term ontology in the computer and information science literature

occurs instead in 1967, in a work on the foundations of data modeling by S. H. Mealy (H.67).
From the scientific point of view, the ontology is, in its first approximation, a table of categories in which every type of entity is captured by some node within a hierarchical tree. While the philosopher-ontologist has only the goal of establishing the truth about the domain in question, in the world of the information systems the ontology is a software (or formal language) designed with a specific set of uses and computational environments. In this field, the ontologies are rapidly developing (from 1990 up to now) thanks to their focus on classification and their abilities in representing the domains, in sharing the knowledge and in keeping separate the domain knowledge from the operational one. Furthermore, they play a key role in the Semantic Web by providing a source of shared and precisely defined terms that can be used for describing the resources. Reasoning over such descriptions is essential for accessibility purposes, automating processes and discovering new knowledge.

2.1.2 Formal ontology definition

One of the modern ontology definition given by (RM03) is reported below. An ontology O is defined as

$$O = \{C, R, A^O\}, \tag{2.1}$$

where:

1. C is a set of elements are called concepts.
2. $R \subseteq (C \times C)$ is a set of elements are called relations. For $r = (c_1, c_2) \in R$, one may write $r(c_1) = c_2$.
3. A^O is a set of axioms on O.

To cope with the lexical level, the notion of a lexicon is introduced. For an ontology structure $O = \{C, R, A^O\}$ a lexicon L is defined as

$$L = \{L^C, L^R, F, G\}, \tag{2.2}$$

where:

1. L^C is a set of elements are called lexical entries for concepts.

2. L^R is a set of elements are called lexical entries for relations.

3. $F \subseteq L^C \times C$ is a reference for concepts such that

   - $F(l_C) = c \in C : (l_C, c) \in F forall l_C \in L^C$,
   - $F^{-1}(c) = l_C \in L^C : (l_C, c) \in F forall c \in C$.

4. $G \subseteq L^R \times R$ is a reference for concepts such that

   - $G(l_R) = r \in R : (l_R, r) \in G \forall l_R \in L^R$,
   - $G^{-1}(r) = l_R \in L^R : (l_R, r) \in G \forall r \in R$.

We note that a hierarchical structure can be explicitly defined in terms of R. Finally, the mapping from elements in the lexicon to elements in the ontology structure corresponds to the notion of an interpretation in declarative semantics (GN87). Such a (semantic) interpretation associates elements of a language to elements of a conceptualisation. Based on the above definitions, an ontology can be formally defined as the pair $< O, L >$ where O is an ontology structure and L is a corresponding lexicon. The ontology structure O plays the role of an explicit specification of a conceptualisation of some domain while the lexicon L provides the agreed vocabulary to communicate about this conceptualisation. The following example may be useful for explaining the above definitions.
Let $O = \{C, R, A^O\}$ be an ontology structure such that $C = \{c_1, c_2\}$ and $R = \{r\}$, where $r(c_1) = c_2$. Suppose that $A^O = \oslash$. Also, let $L = L^C, L^R, F, C$ be a corresponding lexicon such that $L^C = \{Mouse,$ $Inputdevice\}$, $L^R = \{is\_a\}$, $F(Mouse) = c_1$, $F(Inputdevice) = c_2$ and $G(is\_a) = r$.

Figure 1 depicts the elementary ontology described above. The top of the figure shows the ontology structure O. It consists of two concepts c1, c2 and one relation r that relates them. This corresponds to a conceptualisation of a domain of interest without any lexical reference. The latter is provided by the lexicon L depicted at the bottom of the figure. L provides the lexical references for O by means of F and G. F and G map lexical

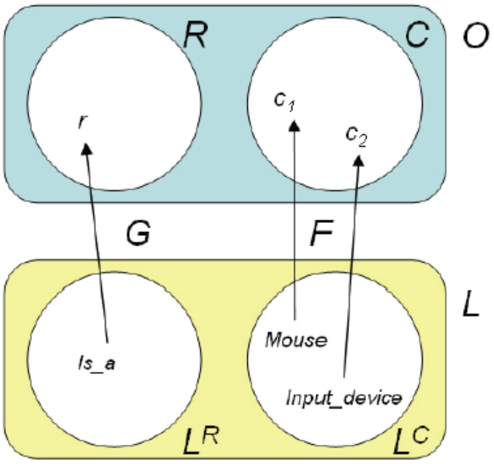


Figure 1: Graphical representation of an ontology.

reference strings to the concepts and relations defined in O, respectively. For instance, for $r \in R$ one may consider using $G^{-1}(r)$ to get the lexical reference, i.e. is_a, corresponding to r and vice versa.

2.1.3 Ontology web language

The Web Ontology Language (OWL) is the current standard provided by the World Wide Web Consortium (W3C) for defining and instantiating Web ontologies. An OWL ontology can include descriptions of classes, properties and their instances. The OWL language provides three increasingly expressive sublanguages:

- OWL Lite supports a classification hierarchy and simple constraint features.
- OWL DL is more expressive than OWL Lite without losing computational completeness (all entailments are guaranteed to be computable) and decidability (all computations will finish in finite time) of reasoning systems. Its name is due to its correspondence with

description logics (BP03), a field of research that has studied a particular decidable fragment of first order logic.

- OWL Full is the most expressive with no computability guarantees.

We focus mainly on the first two variants of OWL because OWL-Full has a nonstandard semantics that makes the language undecidable. OWL comes with several syntaxes, all of which are rather verbose. In this chapter we use the standard DL syntax. For a full DL syntax description, please refer to (BP03). The main building blocks of an OWL ontology are:

- concepts (or classes), representing sets of objects,
- roles (or properties), representing relationships between objects,
- individuals, representing specific objects.

In particular, properties can be used to state relationships between individuals or from individuals to data values. Examples of properties include *hasChild*, *hasRelative*, *hasSibling*, and *hasAge*. The first three can be used to relate an instance of a class *Person* to another instance of the class *Person* (and are thus occurences of $Object\ Property$), and the last (*hasAge*) can be used to relate an instance of the class *Person* to an instance of the datatype *Integer* (and is thus an occurence of $Data\ Property$).
Furthermore they are composed of two parts: intensional and extensional. The former part consists of a $TBox$ and an $RBox$, and contains knowledge about concepts (i.e. classes) and the complex relations between them (i.e. roles). The latter one consists of an $ABox$, containing knowledge about entities and the way they are related to the classes as well as roles from the intensional part. The following is an example of DL formalization (according to the syntax defined in (BP03)) of the previous ontology fragment. We add a property specifying the number of keys of each input device:

$$\begin{gathered} TBox = ((Mouse \sqsubseteq Inputdevice) \sqcap (Inputdevice \sqsubseteq Thing) \\ \sqcup (= 1 hasKeysNumber) \sqsubseteq Inputdevice \\ \sqcup (\forall hasKeysNumber.integer)). \end{gathered}$$

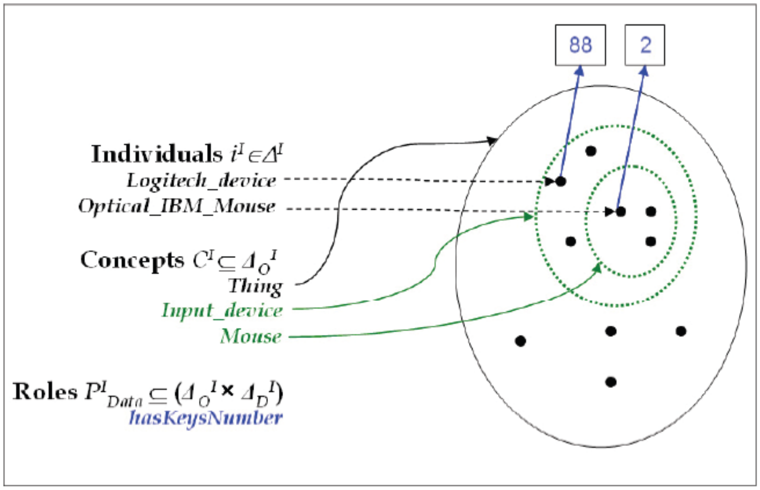

88
2
Individuals $i^I \in \Delta^I$
Logitech_device
Optical_IBM_Mouse
Concepts $C^I \subseteq \Delta_O{}^I$
Thing
Input_device
Mouse
Roles $P^I{}_{Data} \subseteq (\Delta_O{}^I \times \Delta_D{}^I)$
hasKeysNumber

Figure 2: Example of well defined model theoretic semantics.

All the possible ontology instances constitute the $ABox$ which is interlinked with the intensional knowledge. An example of $ABox$ related to the previous $TBox$ is the following:

$$\begin{gathered} ABox = \{Logitech\_device : Input\_device, \\ hasKeysNumber = 88, \\ Optical\_IBM_Mouse : Mouse, \\ hasKeysNumber = 2\} \end{gathered}$$

Given such an ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics.

The semantics of OWL DL is in agreement with DL standard semantics. An example is in figure 2. An interpretation $I = (\Delta^I, \cdot^I)$ is a tuple where Δ^I, the domain of discourse, is the union of two disjoint sets Δ_{O^I}, (the object domain) and Δ_{D^I} (the data domain), and I is the interpretation function that gives meaning to the entities defined in the ontology. I maps each OWL class C to a subset $C^I \subseteq \Delta_{O^I}$, each object property P_{Obj} to a binary relation $P^I_{Obj} \subseteq (\Delta_{O^I} \times \Delta_{O^I})$, each datatype property

P_{Data} to a binary relation $P^I_{Data} \subseteq (\Delta_{O^I} \times \Delta_{D^I})$. The complete definition can be found in the OWL W3C Recommendation [1]. Concepts and relations represent the explicit knowledge of an ontology, nevertheless another kind of knowledge (i.e. the implicit knowledge) can be deduced starting from the known facts. Its existence is implied by or inferred from observable behavior or performance by using the rule inference mechanism. Rules are an extension of the logical core formalism, which can still be interpreted logically. The simplest variant of such rules are expressions of the form $C \Rightarrow D$ where C and D are concepts. The meaning of such a rule is that "if an individual is proved to be an instance of C, then derive that it is also an instance of D". Operationally, the semantics of a finite set R of rules can be described by a forward reasoning process. Starting with an initial knowledge base K, a series of knowledge bases $K^{(0)}, K^{(1)}, ...$ is constructed, where $K^{(0)} = K$ and $K^{(i+1)}$ is obtained from $K^{(i)}$ by adding a new assertion $D(a)$ whenever R contains a rule $C \Rightarrow D$ such that $K^{(i)} \subseteq C(a)$ holds, but $K^{(i)}$ does not contain $D(a)$. Semantic Web Rule Language (SWRL) (Hor04) is the W3C proposal for the rule language based on a combination of the OWL DL and OWL Lite sublanguages with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language [2]. SWRL allows users to write Hornlike rules that can be expressed in terms of OWL concepts and that can reason about OWL individuals (Hor04). According to the SWRL syntax and referring to the previous example of $TBox$ and $ABox$, the inference rule for deducing that an ontology element is an instance of Mouse states that is an Inputdevice and has 2 keys, as reported below:

$$\forall x, y Inputdevice(?x) \wedge hasKeysNumber(?x, ?y) \wedge equalTo(?y, 2) \rightarrow Mouse(?x),$$

where $equalTo(.,.)$ is a SWRL built-in function.

[1] The W3C webpage: http://www.w3c.org/.

[2] Rule Markup Language is specified at http://www.ruleml.org/.

2.2 Bayesian Belief Networks

In this section we provide a short general background of Bayesian Network. In particular, in section 2.2.1 we describe the fundamental concepts of the graphical language used to construct probabilistic networks as well as the rules for reading statements of (conditional) dependence and independence encoded in network structures. Section 2.2.2 presents the uncertainty calculus used in probabilistic networks to represent the numerical counterpart of the graphical structure, namely classical Bayesian probability calculus. We shall see how a basic axiom of probability calculus leads to recursive factorizations of joint probability distributions into products of conditional probability distributions, and how such factorizations along with local statements of conditional independence naturally can be expressed in graphical terms. In section 2.2.3 we shall see how putting the basic notions of section 2.2.1 and 2.2.2 together we get the notion of discrete Bayesian networks and we explain the principal underlying inference in probabilistic networks.

2.2.1 Networks

Probabilistic networks are graphical models of (causal) interactions among a set of variables, where the variables are represented as nodes (also known as vertices) of a graph and the interactions (direct dependences) as directed links (also known as arcs and edges) among the nodes. Any pair of unconnected/nonadjacent nodes of such a graph indicates (conditional) independence between the variables represented by these nodes under particular circumstances that can easily be read from the graph. Hence, probabilistic networks capture a set of (conditional) dependence and independence properties associated with the variables represented in the network. Graphs have proven themselves to be a very intuitive language for representing such dependence and independence statements, and thus provide an excellent language for communicating and discussing dependence and independence relations among problem-domain variables. A large and important class of assumptions about dependence and independence relations expressed in factorized representations of joint proba-

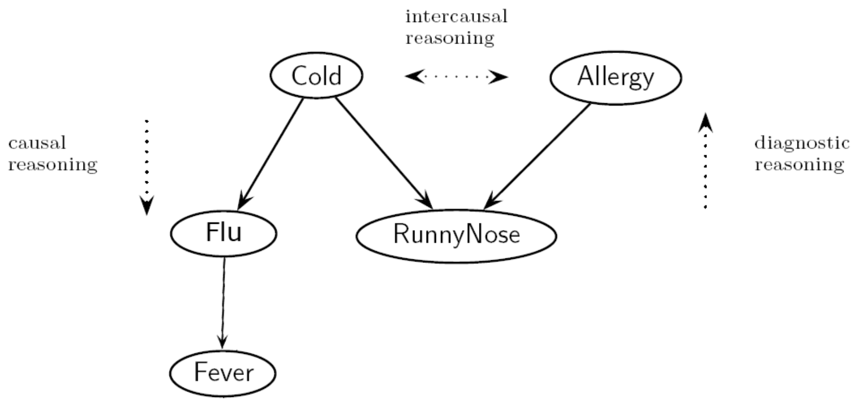


Figure 3: Example of causal network.

bility distributions can be represented very compactly in a class of graphs known as acyclic, directed graphs (DAGs). Chain graphs are a generalization of DAGs, capable of representing a broader class of dependence and independence assumptions (Fry89), (Wer90). The added expressive power comes, however, at the cost of a significant increase in the semantic complexity, making specification of joint probability factors much less intuitive. Thus, despite their expressive power, chain graph models have gained very little popularity as practical models for decision support systems, and we shall therefore focus exclusively on models that factorize according to DAGs.

As indicated above, probabilistic networks is a class of probabilistic models that have gotten their name from the fact that the joint probability distributions represented by these models can be naturally described in graphical terms, where the nodes of a graph (or network) represent variables over which a joint probability distribution is defined and the presence and absence of links represent dependence and independence properties among the variables. Probabilistic networks can be seen as compact representations of fuzzy cause-effect rules that, contrary to ordinary (logical) rule based systems, are capable of performing deductive

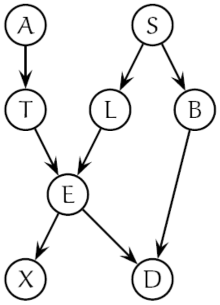


Figure 4: Example of graph.

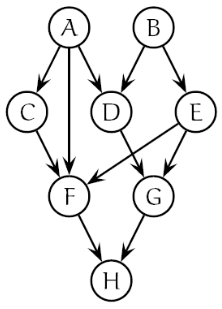
and abductive reasoning as well as intercausal reasoning. Deductive reasoning (sometimes referred to as causal reasoning) follows the direction of the causal links between variables of a model; e.g., knowing that a person has caught a cold we can conclude (with high probability) that the person has fever and a runny nose (see Figure 3). Abductive reasoning (sometimes referred to as diagnostic reasoning) goes against the direction of the causal links; e.g., observing that a person has a runny nose provides supporting evidence for either cold or allergy being the correct diagnosis. The property, however, that sets inference in probabilistic networks apart from other automatic reasoning paradigms is its ability to make intercausal reasoning: Getting evidence that supports solely a single hypothesis (or a subset of hypotheses) automatically leads to decreasing belief in the unsupported, competing hypotheses. This property is often referred to as the explaining away effect. For example, in figure 3, there are two competing causes of runny nose. Observing fever, however, provides strong evidence that cold is the cause of the problem, while our belief in allergy being the cause decreases substantially (i.e., it is explained away by the observation of fever). The ability of probabilistic networks to automatically perform such intercausal inference is a key contribution to their reasoning power. Often the graphical aspect of a probabilistic network is referred to as its qualitative aspect, and the probabilistic, numerical part as its quantitative aspect. This section is devoted to the qualitative aspect of probabilistic networks, which is characterized by DAGs where the nodes represent random variables, decision variables, or utility functions, and where the links represent direct dependences, informational constraints, or they indicate the domains of utility functions. Bayesian networks contain only random variables, and the links represent direct dependences (often, but not necessarily, causal relationships) among the variables. The causal network in Figure 3 shows the qualitative part of a Bayesian network. In the next subsections we introduce some basic graph notation, the evidence concept and the criterion of the d-separation.

Graphs

A graph is a pair $G = (V, E)$, where V is a finite set of distinct nodes and $E \subseteq V \times V$ is a set of links. An ordered pair $(u, v) \in E$ denotes a directed link from node u to node v, and u is said to be a parent of v and v a child of u. The set of parents and children of a node v shall be denoted by $pa(v)$ and $ch(v)$, respectively. As we shall see later, depending on what they represent, nodes are displayed as labelled circles, ovals, or polygons, directed links as arrows, and undirected links as lines. Figure 4 shows a graph with 8 nodes and 8 links (all directed), where, for example, the node labeled E has two parents labeled T and L. The labels of the nodes are referring to (i) the names of the nodes, (ii) the names of the variables represented by the nodes, or (iii) descriptive labels associated with the variables represented by the nodes. Often the intuitive notation $u \overset{G}{\rightarrow} v$ to denote $(u, v) \in E$ is used (or just $u \rightarrow v$ if G is understood). If $(u, v) \in E$ and $(v, u) \in E$, the link between u and v is an undirected link, denoted by $\{u, v\} \in E$ or $u \overset{G}{\leftrightarrow} v$ (or just $u \leftrightarrow v$). We shall use the notation $u \sim v$ to denote that $u \rightarrow v$, $v \leftarrow u$, or $u \leftrightarrow v$. Nodes u and v are said to be connected in G if $u \overset{G}{\sim} v$. If $u \rightarrow v$ and $w \rightarrow v$, then these links are said to meet head-to-head at v. If E does not contain undirected links, then G is a directed graph and if E does not contain directed links, then it is an undirected graph. As mentioned above, we shall not deal with mixed cases of both directed and undirected links. A path $< v_1, ..., v_n >$ is a sequence of distinct nodes such that $v_i \sim v_{i+1}$ for each $i = 1, ..., n - 1$; the length of the path is $n - 1$. The path is a directed path if $v_i \rightarrow v_{i+1}$ for each $i = 1, ..., n-1$; v_i is then an ancestor of v_j and v_j a descendant of v_i for each $j > i$. The set of ancestors and descendants of v are denoted $an(v)$ and $de(v)$, respectively. The set $nd(v) = V/de(v) \cup \{v\}$ are called the non-descendants of v. The ancestral set $An(U) \subseteq V$ of a set $U \subseteq V$ of a graph $G = (V, E)$ is the set of nodes $U \cup \bigcup_{u \in U} an(u)$. A graph $G = (V, E)$ is connected if for any pair $\{u, v\} \subseteq V$ there is a path $< u, ..., v >$ in G. A connected graph $G = (V, E)$ is a polytree (also known as a singly connected graph) if for any pair $\{u, v\} \subseteq V$ there is a unique path $< u, ..., v >$ in G. A directed polytree with only a single orphan node is called a (rooted) tree. A cycle is a path $< v_1, ..., v_n >$ of length greater than two with the exception that $v_1 = v_n$; a directed cycle is defined in the obvious way. A directed graph with no directed cycles is called an acyclic, directed graph or simply a DAG; see figure 4 for an example.

Evidence

A key inference task with a probabilistic network is computation of posterior probabilities of the form $P(x|\varepsilon)$, where, in general, ε is evidence (i.e., information) received from external sources about the (possible) values of a subset of the variables of the network. For a set of discrete evidence variables X, the evidence appears in the form of a likelihood distribution over the states of X. An evidence can be seen as a function, ε_X, for X is a function $\varepsilon_X : dom(X) \rightarrow R^+$. An evidence function that assigns a zero probability to all but one state is often said to provide hard evidence; otherwise, it is said to provide soft evidence. In the next chapters, we shall leave out the hard or soft qualifier, and simply talk about evidence if the distinction is immaterial. Hard evidence on a variable X is also often referred to as instantiation of X or that X has been observed. Note that, as soft evidence is a more general kind of evidence, hard evidence can be considered a special kind of soft evidence.



(1) C and G are d-connected

(2) C and E are d-separated

(3) C and E are d-connected given evidence on G

(4) A and G are d-separated given evidence on D and E

(5) A and G are d-connected given evidence on D

Figure 5: Example of D-separation criterion.

D-separation criterion

The DAG of a Bayesian network model is a compact graphical representation of the dependence and independence properties of the joint probability distribution represented by the model. The causal network in figure 3 shows the five relevant variables (all of which are Boolean; i.e., they have states F and T) and the entailed causal relationships. Notice that all of the links are causal. For example, allergy and cold can cause the fact that Andrea's nose is runny, and if it is cold, Andrea can take flu. To introduce the d-separation criterion, we have to consider each possible basic kind of connection in a DAG. To illustrate these, consider the DAG in figure 5. We see three different kinds of connections:

- **Serial connections**. The general rule for flow of information in serial connections can thus be stated as follows. Information may flow through a serial connection $X \rightarrow Y \rightarrow Z$ unless the state of Y is known.
  - $cold \rightarrow flu \rightarrow fever$
- **Diverging connections**. The general rule for flow of information in serial connections can thus be stated as follows. Information may flow through a diverging connection $X \leftarrow Y \rightarrow Z$ unless the state of Y is known.

  - $flu \leftarrow cold \rightarrow runnynose$

- **Converging connections**. The general rule for flow of information in serial connections can thus be stated as follows. Information may flow through a converging connection $X \rightarrow Y \leftarrow Z$ if evidence on Y or one of its descendants is available.

  - $cold \rightarrow runnynose \leftarrow allergy$

The previous propositions can be summarized into a rule known as d-separation (Pea88):

Definition 1 ***(d-Separation)***. *A path $\pi = < u, ..., v >$ in a DAG, $G = (V, E)$, is said to be blocked by $S \subseteq V$ if π contains a node w such that either*

- *$w \in S$ and the links of π do not meet head-to-head at w, or*

- *$w \notin S$, $de(w) \cap S = \oslash$, and the links of π meet head-to-head at w.*

For three (not necessarily disjoint) subsets A, B, S of V, A and B are said to be d-separated if all paths between A and B are blocked by S.

We can make definition 1 operational through a focus on nodes or through a focus on connections. Let $G = (V, E)$ be a DAG of a causal network and let $H_\varepsilon \subseteq S_\varepsilon \subseteq V$ be subsets of nodes representing, respectively, variables with hard evidence and variables with soft evidence on them. Assume that we wish to answer the question: "Are nodes v_1 and v_n d-separated in G under evidence scenario S_ε?". Now, using a nodes approach to d-separation, we can answer the question as follows. If for any path $\pi = < v_1, ..., v_n >$ between v_1 and v_n and for each $i = 2, ...n - 1$ either

- $v_i \in H_\varepsilon$ and the connection $v_{i-1} \sim v_i \sim v_{i+1}$ is serial or diverging, or

- $(\{v_i\} \cup de(v_i)) \cap S_\varepsilon = \oslash$ and $v_{i-1} \rightarrow v_i \leftarrow v_{i+1}$,

then v_1 and v_n are d-separated given S_ε; otherwise, they are d-connected given S_ε. Often, however, it can be more intuitive to think in terms of flow of information, in which case a connections (or flow-of-information) approach to d-separation might be more natural. If for some path $<$

$v_1, ..., v_n >$ between v_1 and v_n and for each $i = 2, ...n-1$ the connection $v_{i-1} \sim v_i \sim v_{i+1}$ allows flow of information from v_{i-1} to v_{i+1}, then v_1 and v_n are d-connected; otherwise, they are d-separated.
We show a d-separation example. Consider the problem of figuring out whether variables C and G are d-separated in the DAG in figure 5; that is, are C and G independent when no evidence about any of the other variables is available? Using the flow-of-information approach, we first find that there is a diverging connection $C \leftarrow A \rightarrow D$ allowing flow of information from C to D via A. Second, there is a serial connection $A \rightarrow D \rightarrow G$ allowing flow of information from A to G via D. So, information can thus be transmitted from C to G via A and D, meaning that C and G are not d-separated (i.e., they are d-connected). C and E, on the other hand, are d-separated, since each path from C to E contains a converging connection, and since no evidence is available, each such connection will not allow flow of information. Given evidence on one or more of the variables in the set $\{D, F, G, H\}$, C and E will, however, become d-connected. For example, evidence on H will allow the converging connection $D \rightarrow G \leftarrow E$ to transmit information from D to E via G, as H is a child of G. Then information may be transmitted from C to E via the diverging connection $C \leftarrow A \rightarrow D$ and the converging connection $D \rightarrow G \leftarrow E$.

2.2.2 Probabilities

As mentioned in section 2.2, probabilistic networks have a qualitative aspect and a corresponding quantitative aspect, where the qualitative aspect is given by a graphical structure in the form of an acyclic, directed graph (DAG) that represents the (conditional) dependence and independence properties of a joint probability distribution defined over a set of variables that are indexed by the nodes of the DAG. The fact that the structure of a probabilistic network can be characterized as a DAG derives from basic axioms of probability calculus leading to recursive factorization of a joint probability distribution into a product of lower-dimensional conditional probability distributions. First, any joint probability distri-

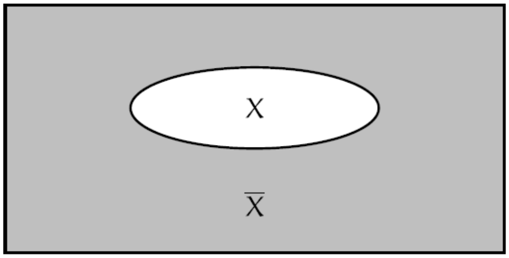


Figure 6: Universe of elements $U = X \cup \overline{X}$.

bution can be decomposed (or factorized) into a product of conditional distributions of different dimensionality, where the dimensionality of the largest distribution is identical to the dimensionality of the joint distribution. Second, statements of local conditional independences manifest themselves as reductions of dimensionalities of some of the conditional probability distributions. Collectively, these two facts give rise to a DAG structure. In fact, a joint probability distribution, P, can be decomposed recursively in this fashion if and only if there is a DAG that correctly represents the (conditional) dependence and independence properties of P. This means that a set of conditional probability distributions specified according to a DAG, $G = (V, E)$, (i.e., a distribution $P(A|pa(A))$ for each $A \in V$) define a joint probability distribution. Therefore, a probabilistic network model can be specified either through direct specification of a joint probability distribution, or through a DAG (typically) expressing cause-effect relations and a set of corresponding conditional probability distributions. Obviously, a model is almost always specified in the latter fashion. This section presents some basic axioms of probability calculus from which the famous Bayes rule follows as well as the chain rule for decomposing a joint probability distribution into a product of conditional distributions. We shall also present the fundamental operations needed to perform inference in probabilistic networks.

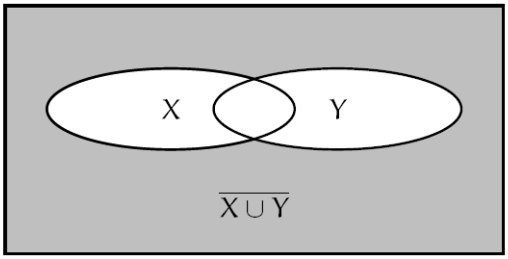


Figure 7: Probability of the union of events.

Definition of probability

Let us start out by defining informally what we mean by probability. Apart from introducing the notion of probability, this will also provide some intuition behind the three basic axioms presented so far. Consider a (discrete) universe, U, of elements, and let $X \subseteq U$. Denote by $\overline{X} = U/X$ the complement of X. Figure 6 on the facing page illustrates U, where we imagine that the area covered by X, is proportional to the number of elements that it contains. The chance that an element sampled randomly from U belongs to X defines the probability that the element belongs to X, and is denoted by $P(X)$. Note that $P(X)$ can informally be regarded as the relative area occupied by X in U. That is, $P(X)$ is a number between 0 and 1. Suppose now that $U = X \cup Y \cup \overline{X \cup Y}$ as in figure 7. The probability that a random element from U belongs to $X \cup Y$ is defined as $P(X \cup Y) = P(X) + P(Y) - P(X \cap Y)$. Again, we can interpret $P(X \cup Y)$ as the relative area covered jointly by X and Y. So, if X and Y are disjoint (i.e., $X \cap Y = \oslash$), then $P(X \cup Y) = P(X) + P(Y)$. The conjunctive form $P(X \cap Y)$ is often written as $P(X, Y)$.

Consider figure 8 and assume that we know that a random element from U belongs to Y. The probability that it also belongs to X is then calculated as the ratio $\frac{P(X \cap Y)}{P(Y)}$. Again, to understand this definition, it may help to consider $P(X \cap Y)$ and $P(Y)$ as relative areas of U. We shall use the notation $P(X|Y)$ to denote this ratio, where $|$ reads ”given that

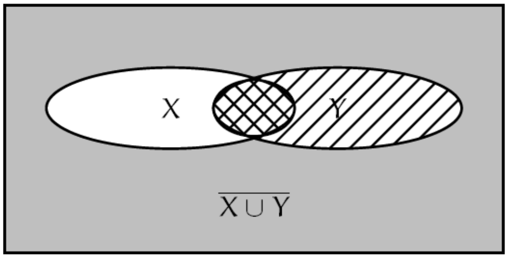


Figure 8: Conditional Probability of X given Y.

we know" or simply "given". Thus, we have

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

where $P(X|Y)$ reads "the conditional probability of X given Y".

Events

The language of probabilities consists of statements (propositions) about probabilities of events. As an example, consider the event, a, that a sample a universe $U = X \cup \overline{X}$ happens to belong to X. The probability of an event a is denoted $P(a)$. In general, an event can be considered as the outcome of an experiment (e.g., a coin flip), a particular observation of the value of a variable (or set of variables), etc. As a probabilistic network define a probability distribution over a set of variables, V, in our context an event is a configuration, $x \in dom(X)$, (i.e., a vector of values) of a subset of variables $X \subseteq V$.

Conditional probability

The basic concept in the Bayesian treatment of uncertainty is that of conditional probability: Given event b, the conditional probability of event a is x, written as

$$P(a|b) = x. \tag{2.4}$$

This means that if event b is true and everything else known is irrelevant for event a, then the probability of event a is x. For example, referring to figure 6, assume that Andrea's nose is runny in eight of every ten case (A), when the weather is cold (E), but Andrea has no allergies (B). This fact would then be expressed as the conditional probability $P(A = yes|B = no, E = yes) = 0.8$.

Axioms

The following three axioms provide the basis for Bayesian probability calculus, and summarizes the considerations of section 2.2.2.
Axiom 1. For any event, a, $0 \leq P(a) \leq 1$, with $P(a) = 1$ if and only if a occurs with certainty.
Axiom 2. For any two mutually exclusive events a and b the probability that either a or b occur is

$$P(a \vee b) = P(a) + P(b). \tag{2.5}$$

In general, if events $a_1, ..., a_n$ are pairwise mutually exclusive, then

$$P(\bigcup_i^n a_i) = P(a_1) + ... + P(a_n) = \sum_i^n P(a_i). \tag{2.6}$$

Axiom 3. For any two events a and b the probability that both a and b occur is

$$P(a \wedge b) \equiv P(a, b) = P(b|a)P(a) = P(a|b)P(b). \tag{2.7}$$

$P(a, b)$ is called the joint probability of the events a and b. Axiom 1 simply says that a probability is a non-negative real number less than or equal to 1, and that it equals 1 if and only if the associated event happens for sure. Axiom 2 says that if two events cannot co-occur, then the probability that either one of them occurs equals the sum of the probabilities of their individual occurrences. Axiom 3 is sometimes referred to as the fundamental rule of probability calculus. The axiom says that the probability of the co-occurrence of two events, a and b can be computed as the product of the probability of event a (b) occurring given that event b (a) has already occurred.

Probability Distributions for Variables

Probabilistic networks are defined over a (finite) set of variables, each of which represents a finite set of exhaustive and mutually exclusive states (or events). Thus, (conditional) probability distributions for variables (i.e., over exhaustive sets of mutually exclusive events) play a central role in probabilistic networks. If X is a (random) variable with domain $dom(X) = (x_1, ..., x_{||X||})$, then $P(X)$ denotes a probability distribution (i.e., a vector of probabilities summing to 1), where

$$P(X) = (P(X = x_1), ..., P(X = x_{||X||})). \tag{2.8}$$

We use $P(x)$ as shorthand for $P(X = x)$, etc. If the probability distribution for a variable Y is given conditionally w.r.t. a variable (or set of variables) X, then we shall use the notation $P(Y|X)$. That is, for each possible value (state), $x \in dom(X)$, we have a probability distribution $P(Y|X = x)$. We write $P(Y|x)$.

Rule of Total Probability

Let $P(X, Y)$ be a joint probability distribution for two variables X and Y with $dom(X) = x_1, ..., x_m$ and $dom(Y) = y_1, ..., y_n$. Using the fact that $dom(X)$ and $dom(Y)$ are exhaustive sets of mutually exclusive states of X and Y, respectively, Axiom 2 defined so far gives us the rule of total probability:

$$\forall i : P(x_i) = P(x_i, y_1) + ... + P(x_i, y_n) = \sum_{j=1}^{n} P(x_i, y_j). \tag{2.9}$$

Using 2.9, we can calculate $P(X)$ from $P(X, Y)$:

$$P(X) = (\sum_{j=1}^{n} P(x_1, y_j), ..., \sum_{j=1}^{n} P(x_m, y_j)). \tag{2.10}$$

In a more compact notation, we may write

$$P(X) = \sum_{j=1}^{n} P(X, y_j), \tag{2.11}$$

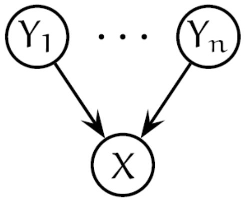


Figure 9: Example of BN.

or even shorter as

$$P(X) = \sum_{Y} P(X, Y), \tag{2.12}$$

denoting the fact that we sum over all indices of Y. We shall henceforth refer to the operation in 2.12 as marginalization or projection. Also, we sometimes refer to this operation as marginalizing out Y of $P(X, Y)$ or eliminating Y from $P(X, Y)$.

Graphical Representation

The conditional probability distributions of probabilistic networks are of the form $P(X|Y)$, where X is a single variable and Y is a (possibly empty) set of variables. X and Y are sometimes called the head and the tail, respectively, of $P(X|Y)$. If $Y = \oslash$ (i.e., the empty set), $P(X|Y)$ is often called a marginal probability distribution and is then written as $P(X)$. This relation between X and $Y = \{Y_1, ..., Y_n\}$ can be represented graphically as the DAG illustrated in figure 9, where the child node is labelled X and the parent nodes are labelled $Y1$, $Y2$, etc.

Fundamental Rule and Bayes Rule

Generalizing Axiom 3 to arbitrary (random) variables X and Y we get the fundamental rule of probability calculus:

$$P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X). \tag{2.13}$$

Bayes rule follows immediately:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}. \tag{2.14}$$

Using Axiom 3 and the rule of total probability, 2.14 can be rewritten like

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X|Y=y_1)P(Y=y_1) + \;\; + P(X|Y=y_n)P(Y=y_{||Y||})}. \tag{2.15}$$

We often write Bayes rule as

$$P(Y|X) \propto P(X|Y)P(Y). \tag{2.16}$$

read as "$P(Y|X)$ is proportional to $P(X|Y)P(Y)$". Note that the proportionality factor $P(X)^{-1}$ is in fact a vector of proportionality constants, one for each state of X, determined in a normalization operation. Division by zero in 2.14 is not a problem as we define 0/0 = 0, since for

$$P(x_i) = \sum_j P(x_i|y_j)P(y_j) \tag{2.17}$$

to be zero at least one of $P(x_i|y_j)$ and $P(y_j)$ must be zero for each j, and if this is the case then both the numerator $P(x_i|y_j)P(y_j)$, and the denominator $P(x_i)$, of 2.14 will be zero.

Independence

A variable X is independent of another variable Y with respect to a probability distribution P if

$$P(x|y) = P(x), \forall x \in dom(X), \forall y \in dom(Y). \tag{2.18}$$

We express this property as $X \bowtie_P Y$, or simply as $X \bowtie Y$ when P is obvious from the context. Symmetry of independence (i.e., $X \bowtie Y \equiv Y \bowtie X$) can be verified from Bayes rule:

$$P(x) = P(x|y) = \frac{P(y|x)P(x)}{P(y)} \Leftrightarrow P(y) = P(y|x). \tag{2.19}$$

The statement $X \bowtie Y$ is often referred to as marginal independence between X and Y. If $X \bowtie Y$, then from the fundamental rule 2.13 and 2.18 we get that

$$P(X, Y) = P(X|Y)P(Y) = P(X)P(Y) \tag{2.20}$$

and, in general, whenever $X_1, ..., X_n$ are pairwise independent random variables their joint distribution equals the product of the marginals:

$$P(X_1, ..., X_n) = \prod_i P(X_i). \tag{2.21}$$

A variable X is conditionally independent of Y given Z (with respect to a probability distribution P) if

$$P(x|y, z) = P(x|z), \forall x \in dom(X), \forall y \in dom(Y), \forall z \in dom(Z). \tag{2.22}$$

The conditional independence statement expressed in 2.22 is indicated as $X \bowtie Y|Z$ in our standard notation. Again, from the fundamental rule and 2.22 we get

$$\begin{aligned} P(X, Y, Z) &= P(X|Y, Z)P(Y, Z) \\ &= P(X|Y, Z)P(Y|Z)P(Z) \\ &= P(X|Z)P(Y|Z)P(Z). \end{aligned} \tag{2.23}$$

Chain rule

For a probability distribution, $P(X)$, over a set of variables $X = \{X_1, ..., X_n\}$, we can use the fundamental rule repetitively to decompose it into a product of conditional probability distributions:

$$\begin{aligned} P(X) &= P(X_1|X_2, ..., X_n)P(X_2, ..., X_n) \\ &= P(X_1|X_2, ..., X_n)P(X_2|X_3, ..., X_n) \cdots P(X_{n-1}|X_n)P(X_n) \\ &= \prod_{i=1}^{n} P(X_i|X_{i+1}, ..., X_n). \end{aligned} \tag{2.24}$$

Notice that the actual conditional distributions that comprise the factors of the decomposition are determined by the order in which we select the head variables of the conditional distributions. Thus, there are n factorial different factorizations of $P(X)$, and to each factorization corresponds a unique DAG, but all of these DAGs are equivalent in terms of dependence and independence properties, as they are all complete graphs, and hence represent no independence statements.

2.2.3 Probabilistic Networks and Inference

A probabilistic interaction model on a set of random variables may be represented as a joint probability distribution. Considering the case where random variables are discrete, it is obvious that the size of the joint probability distribution will grow exponentially with the number of variables as the joint distribution must contain one probability for each configuration of the random variables. Therefore, we need a more compact representation for reasoning about large and complex systems involving a large number of variables. To the purpose of facilitating an efficient representation of a large and complex domain with many random variables, the framework of Bayesian networks uses a graphical representation to encode dependence and independence relations among the random variables. The dependence and independence relations induce a compact representation of the joint probability distribution. By representing the dependence and independence relations of a domain explicitly in a graph, a compact representation of the dependence and independence relations is obtained.

Discrete Bayesian Networks

A (discrete) Bayesian network, $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$, over variables, $\mathcal{X}$ consists of an acyclic, directed graph $G = (V, E)$ and a set of conditional probability distributions $\mathcal{P}$. Each node v in G corresponds one-to-one with a discrete random variable $X_v \in X$ with a finite set of mutually exclusive states. The directed links $E \subseteq V \times V$ of G specify assumptions of conditional dependence and independence between random variables according to the d-separation criterion. There is a conditional probability distribution, $P(X_v|X_{pa(v)}) \in \mathcal{P}$, for each variable $X_v \in X$. The variables represented by the parents, $pa(v)$, of $v \in V$ in $G = (V, E)$ are sometimes called the conditioning variables of X_v - the conditioned variable.

Definition 2 *A (discrete) Bayesian network $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$ consists of*

- *A DAG $G = (V, E)$ with nodes $V = \{v_1, ..., v_n\}$ and directed links E.*
- *A set of discrete random variables, $\mathcal{X}$, associated to the nodes of G.*
- *A set of conditional probability distributions, P, containing one distribution, $P(X_v|X_{pa(v)})$, for each random variable $X_v \in \mathcal{X}$.*

A Bayesian network encodes a joint probability distribution over a set of random variables, $\mathcal{X}$, of a problem domain. The set of conditional probability distributions, $\mathcal{P}$, specifies a multiplicative factorization of the joint probability distribution over $\mathcal{X}$ as represented by the chain rule of Bayesian networks:

$$P(\mathcal{X}) = \prod_{v \in V} P(X_v|X_{pa(v)}). \tag{2.25}$$

Even though the joint probability distribution specified by a Bayesian network is defined in terms of conditional independence, a Bayesian network is most often constructed using the notion of cause-effect relations. In practice, cause-effect relations between entities of a problem domain can be represented in a Bayesian network by using a graph of nodes representing random variables and links representing cause-effect relations between the entities. Usually, the construction of a Bayesian network (or any probabilistic network for that matter) proceeds according to an iterative procedure where the set of nodes and their states, and the set of links are updated iteratively as the model becomes more and more refined. To solve a Bayesian network $\mathcal{N} = (\mathcal{X}, G, \mathcal{P})$ is to compute all posterior marginals given a set of evidence ε, i.e., $P(X|\varepsilon)$ for all $X \in \mathcal{X}$. If the evidence set is empty, i.e., $\varepsilon = \oslash$, then the task is to compute all prior marginals, i.e., $P(X)$ for all $X \in \mathcal{X}$.

Inference schemas

In general, probabilistic inference is an NP-hard task (Coo90). Even approximate probabilistic inference is NP-hard (Dag93). For certain classes

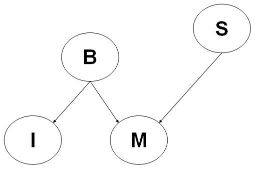


Figure 10: A simple Bayesian network.

of Bayesian network models, the complexity of probabilistic inference is polynomial or even linear in the number of variables in the network. The complexity is polynomial when the graph of the Bayesian network is a polytree (Kim83), (Pea88) (a directed graph G is called a polytree if its underlying undirected graph is singly connected), while it is linear when the graph of the Bayesian network is a tree. The most critical problem related to the efficiency of the inference process is that of finding the optimal order in which to perform the computations. The inference task is, in principle, solved by performing a sequence of multiplications and additions. Three important inference schemas over Bayesian networks exist. In the following we show you the basic idea behind each inference schema. The first schema is the $causal\ inference$ one. In general, it permits to compute the probability of an effect E given one of its cause C_1. The main steps are re-writing the conditional probability of E given the evidence C_1 in terms of all the probabilities of E and all of its parents (which are not part of the evidence C_1), given the evidence. For example, referring to figure 10, in order to compute $P(M|S)$ we have:

$$P(M|S) = P(M|B,S)P(B) + P(M|\overline{B},S)P(\overline{B}) \tag{2.26}$$

The second schema is the $diagnostic\ inference$ one. It permits to compute the probability of a cause C_1 given its effect E. By applying Bayes rule, the diagnostic reasoning is transformed into a causal reasoning for less than a normalisation factor (see section 4.3). For example, in order to compute $P(S|M)$ we have:

$$P(S|M) = \frac{P(M|S)P(S)}{P(M)} \tag{2.27}$$

Note that this is the application of the Bayes formula, and $P(M|S)$ is computable by applying a causal inference schema.
The last schema is the $explaining\ away$ one. It is a combination of the previous schemas. It uses a step of causal inference within a diagnostic process. For example, for computing $P(S|B,M)$, we have to apply Bayes rule:

$$P(S|B,M) = \frac{P(M,B|S)P(S)}{P(B,M)} \tag{2.28}$$

By using the definition of conditional probability we have:

$$\frac{P(M,B|S)P(S)}{P(B,M)} = \frac{P(M|B,S)P(B|S)P(S)}{P(B,M)} \tag{2.29}$$

Finally, depending on the structure of the network in figure 10, we have:

$$\frac{P(M|B,S)P(B|S)P(S)}{P(B,M)} = \frac{P(M|B,S)P(B)P(S)}{P(B,M)} \tag{2.30}$$

The inference schemas presented here, will be discussed in large in Chapter 4.

2.2.4 Related works

Our natural choice for dealing with uncertainty, is the probability theory, in particular the bayesian one. We are attracted to bayesian networks in this work also for the structural similarity between the DAG of a bayesian network and the RDF graph of an ontology: both of them are directed graphs, and direct correspondence exists between many nodes and arcs in the two graphs.

As in (DP04a), (DP04b), (DP05b), (DP05a), we treat the probability as the chance of an instance of belonging to a particular concept class, given the current knowledge. Note that the definition of the concept class itself is precise. In those works the authors represent the probability concept within OWL, that is, they develop a framework which augments and supplements OWL for representing and reasoning with uncertainty based on Bayesian networks. Their research involves developing methods for representing probabilities in OWL statements, and modifies Bayesian networks to satisfy given general probabilistic constraints by changing conditional probability tables only. They augment OWL semantics to allow probabilistic information to be represented via additional markups. The result would be a probabilistic annotated ontology that could then be translated to a Bayesian network (BN). Such a translation would be based on a set of translation rules that would rely on the probabilistic information attached to individual concepts and properties within the annotated ontology. BNs provide an elegant mathematical structure for modeling complex relationships among hypotheses while keeping a relatively simple visualization of these relationships. The technique of representing an ontology by means of a bayesian network is used for having a new knowledge base enriched with uncertainty, over which making inference and probabilistic reasoning. The main difference with (DP04a), (DP04b), (DP05b), (DP05a), is that we deal with object properties. By using a two-levels Bayesian network we can specify a probability distribution for both $is - a$ relationships and $object\ properties$. Consequently, the reasoning is not limited to concept satisfiability, concept overlapping, or concept subsumption, but it also extended to the existence of relationships among classes which are no subsumptions. A path over a two-levels Bayesian network defined by a query can involve object properties with specific values of their domains and their codomains. Domains and codomains are Bayesian networks each of which has a probability distribution of the innstances w.r.t. the $is - a$ relation.

In (CK05) an extension of OWL is presented with full first-order expressiveness by using Multi Entity Bayesian Networks (Las05), that enables OWL ontologies to represent complex Bayesian probabilistic models, in a way that is flexible enough to be used by diverse Bayesian probabilistic tools based on different probabilistic technologies. All those approaches suppose to modify or re-arrange the original knowledge base for dealing with uncertainty, by introducing new relations, or using non-classical bayesian networks. The spirit of our work differs from the other ones exactly because it does not require to modify or extend the original knowledge base, in terms of both schema and instances. UnBBayes-

MEBN (Cos08) implements the PR-OWL probabilistic ontology language (CK05). Knowledge is expressed in MEBN as a collection of MEBN fragments (MFrags) organized into MEBN Theories (MTheories). An MFrag represents a conditional probability distribution of the instances of its resident random variables (RVs) given the values of instances of their parents in the fragment graphs and given the context constraints. RVs are graphically represented in an MFrag either as resident nodes, which have distributions defined in their home fragment, or as input nodes, which have distributions defined elsewhere. The joint distribution is specified by means of the local distributions together with the conditional independence relationships implied by the fragment graphs. Our method aims to use a classical Bayesian theory. Each ontology can be decomposed into taxonomies or more in general into concepts domain (each taxonomy can have more than one root classes). Each concept domain represents a probability distribution of the instances w.r.t. the $is - a$ relationships. In this sense, knowledge is expressed in our work as a collection of domain concepts (multi valued random variables) connected among them via relationships ($object\ properties$). The context, that is the situation in which the values of the random variables are evaluated, is determined by each query. The relationships involved in each query and their values establish constraints under which the local distributions of the nodes involved in the inference path, have to be dinamically computed. Well-known inference schemas can be applyed, such as botto-up one, top-down one, and explaining away one over classical Bayesian networks.

Another important aspect of our work is that the concept of probability is derived from the knowledge base, and it is not directly represented within the ontology. Authors in (CK05), start from this perspective, that is to define a probabilistic ontology to represent the uncertainty explicitely in the ontology. They use the following definition of a probabilistic ontology: a probabilistic ontology (PO) is an explicit, formal knowledge representation that expresses knowledge about a domain of application. This includes: (a) types of entities that exist in the domain; (b) properties of those entities; (c) relationships among entities; (d) processes and events that happen with those entities; (e) statistical regularities that characterize the domain; (f) inconclusive, ambiguous, incomplete, unreliable, and dissonant knowledge related to entities of the domain; and (g) uncertainty about all the above forms of knowledge; where the term entity refers to any concept (real or fictitious, concrete or abstract) that can be described and reasoned about within the domain of application. PR-OWL was developed as an extension enabling OWL ontologies to represent complex Bayesian probabilistic models in a way that is flexible enough to be used by diverse Bayesian probabilistic tools based on different probabilistic technologies (e.g. PRMs, BNs, etc.). More specifically, PR-OWL is an upper ontology (i.e. an ontology that represents fundamental concepts that cross disciplines and applications) for probabilistic systems that is expressive enough to represent even the most complex probabilistic models. It consists of a set of classes, subclasses and properties that collectively form a framework for building POs. Currently, the first step toward building a probabilistic ontology as defined above is to import the PR-OWL ontology, and start constructing the domain-specific concepts, using the PR-OWL definitions to represent uncertainty about their attributes and relationships. In this sense we remark that our query language can deal with all the existing ontologies, and it is no requiring for any kind of probability annotation, or modify to the ontology.

Other works, trying to extend description logics with Bayesian networks, are P-CLASSIC (Kol97) that extends CLASSIC, PTDL (Yel99) that extends TDL (Tiny Description Logic with only Conjunction and Role Quantification operators), OWL-QM (PA05) that extends OWL to support the representation of probabilistic relational models (PRMs) (Get02), and (HH04), (HH05) which uses BNs to model the degree of subsumption for ontologies encoded in RDF(S).
In general, there are two different directions of research related to handling uncertainty. The first is trying to extend the current ontology representation formalism with uncertainty reasoning, the second is to represent probabilistic information using an OWL or RDF(S) ontology. Earliest works have tried to add probabilities into full first-order logic (Bac90), (Hal90), by defining a syntax and a semantics, but the logic was highly undecidable just as pure first-order logic. An alternative direction is to

integrate probabilities into less expressive subsets of first-order logic such as rule-based (for example, probabilistic horn abduction (Poo93)) or object-centered (for example, probabilistic description logics (Hei94), (Jae94), (GL02), (HS05), (NF04), (DP05a), (Str04), (CK05), (Str05), (PH05)) systems. On the other side, to deal with vague and imprecise knowledge, research in extending description logics with fuzzy reasoning has gained some attention. Interested readers can may refer to (AH05), (MD05), (Sto05).
We remark that in our work the technique of representing an ontology by means of a bayesian network is used for having a new knowledge base enriched with uncertainty, over which making inference and probabilistic reasoning. Our final goal in fact, is to provide a query language for answering queries involving probabilities, by integrating ontologies with typical features of bayesian networks.

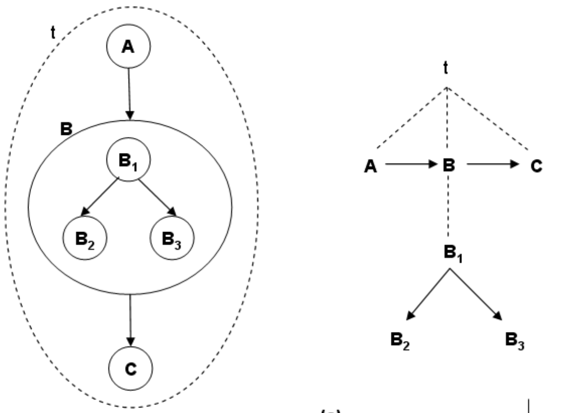
Chapter 3

Compiling Ontologies into Bayesian Networks

3.1 Introduction

In this chapter we introduce a specific structure called two-levels Bayesian Network ($2lBN$ in the following), in order to represent an ontology over which making inference. The main definitions of $2lBN$ are based on the Hierarchical Bayesian Networks features (Fla00), (Gyf04). $2lBN$s are very similar to Bayesian Networks in that they represent probabilistic dependencies between variables as a direct acyclic graph, where each node of the graph corresponds to a random variable and is quantified by the conditional probability of that variable given the values of its parents in the graph. What extends $2lBN$ is that each node can contain, in turn, to a Bayesian Network. So, a node can represent a complex hierarchical domain, rather than a simple event, as well as each classical Bayesian node has to represent. Then, each arc binding nodes among each other, represent the relationships among values of domains that each node codes.
Then, we will define a compiling process in order to represent ontological knowledge by means of Bayesian Networks. In particular, we will establish requirements and hypotheses that an ontology has to satisfy for representing itself as Bayesian Network. We will discuss of the compiling

process about the derivation of both the structural part and the probabilistic part of the Bayesian Network. Finally, we will show some example of ontology compiling process.



(a)

P(A)	a1	a2
	0.1	0.9

$P(B_1 \mid A)$	b_1^1	b_1^2	b_1^3
a1	0.3	0.4	0.3
a2	0.1	0.6	0.3

$P(C \mid B_1, B_2, B_3)$	c1	c2	c3
$b_1^1 b_2^1 b_3^1$	0.3	0.4	0.3
$b_1^1 b_2^2 b_3^2$	0.1	0.6	0.3
$b_1^1 b_2^1 b_3^1$	0.3	0.4	0.3
$b_1^1 b_2^2 b_3^2$	0.1	0.4	0.5
$b_1^2 b_2^1 b_3^1$	0.3	0.1	0.6
$b_1^2 b_2^1 b_3^2$	0.1	0.2	0.7
$b_1^2 b_2^2 b_3^1$	0.1	0.8	0.1
$b_1^2 b_2^2 b_3^2$	0.3	0.1	0.6
$b_1^3 b_2^1 b_3^1$	0.1	0.4	0.5
$b_1^3 b_2^1 b_3^2$	0.1	0.5	0.4
$b_1^3 b_2^2 b_3^1$	0.1	0.7	0.2
$b_1^3 b_2^2 b_3^2$	0.3	0.3	0.4

$P(B_2 \mid B_1, A)$	b_2^1	b_2^2
b_1^1a1	0.3	0.7
b_1^2a2	0.1	0.9
b_1^3a1	0.3	0.7
b_1^1a2	0.5	0.5
b_1^2a1	0.2	0.8
b_1^3a2	0.4	0.6

$P(B_3 \mid B_1, A)$	b_3^1	b_3^2
b_1^1a1	0.9	0.1
b_1^2a2	0.8	0.2
b_1^3a1	0.3	0.7
b_1^1a2	0.6	0.4
b_1^2a1	0.1	0.9
b_1^3a2	0.4	0.6

(b)

Figure 11: (a) Example of 2lBN-structure. (b) Example of 2lBN-probabilistic part.

3.2 Two-levels Bayesian Network

Intuitively, $2lBN$s are a generalisation of standard Bayesian Networks where, at the higher level, a node in the network can be a Bayesian Network coding hierarchical domains. Figure 11 shows a simple example of $2lBN$, where B is a composite node representing an hierarchical domain. This allows the random variables of the network to represent the values of the domain they code, at the lower level. It means that within a single node, there may also be links between components, representing probabilistic dependencies among parts of the lower level structure. $2lBN$s encode conditional probability dependencies the same way as standard Bayesian Networks.

In general by using $2lBN$s is possible to express dependencies in struc-

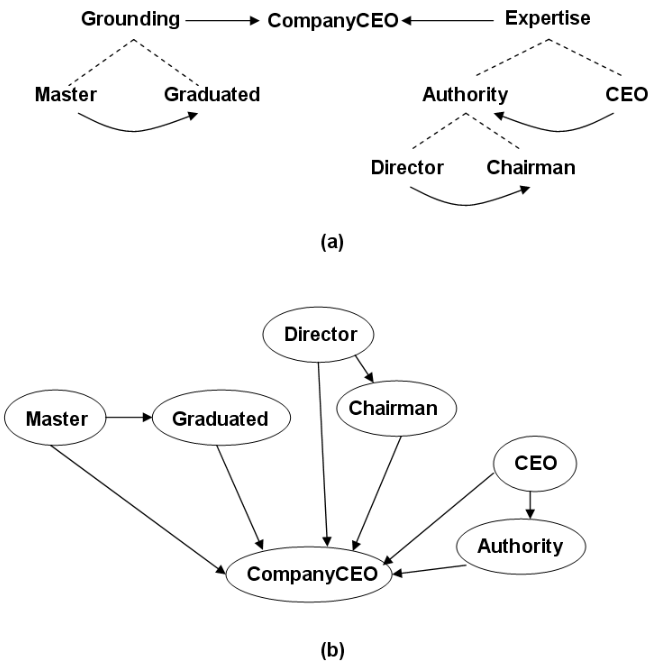


Figure 12: (a) 2lBN Example. (b) Classical Bayesian Network.

tured domains. Then, our idea is to use this representation for capturing the ontological knowledge about domains, and make inference over it. Just to give you a general idea of how to express dependencies in struc-

tured domains, we describe a very simple example. Consider a random variable *companyCEO* that expresses the probability that a particular person is suitable to be the CEO of a company. That decision is influenced by the *expertise* of a person and by his *grounding* (we suppose grounding and expertise to be independent each other). Furthermore we assume that grounding is a couple of random variables [*master*, *graduated*] and expertise another couple of random variables [*authority*, *CEO*] where authority consists itself of the pair [*director*, *chairman*]. A hypothetical configuration of the probabilistic dependencies between those variables is illustrated as a $2lBN$ in figure 12(a). We can observe that the $2lBN$ structure is much more informative than a standard Bayesian Network mapping the same independencies, shown in figure 12(b). Additionally, the assumption that the components of expertise and grounding are independent each other is shown explicitely in the structure. It can easily be extended (adding more components inside the expertise composite node) or refined (transforming a leaf node into a composite one) on the basis, for example, of a taxonomical knowledge provided by ontologies.
Now we give some simple definitions related to the notation that we will use in the rest of the chapter.

Definition 3 ***(High level nodes and relations)*** *Each node representing a specific domain by means of a Bayesian Network is called High Level Node (HLN). Each arc binding HLNs among each other, is called High Level Relation (HLR).*

Definition 4 ***(Low level nodes and relations)*** *Each node within an HLN, that represents a specific value of the domain represented by that HLN, is called Low Level Node (LLN). Each arc binding LLNs among each other, is called Low Level Relation (LLR).*

Definition 5 ***(Domain type)*** *A domain type τ is the type of each HLN, containing a Bayesian Network representing all the values of that domain. Each of them has type τ.*

We specify that each HLR is labelled, because it refers to the name of the specific relation that represents, among HLNs with different domain types. It is not necessary to label LLRs, because they represent $is-a$ relationships among domains values, within specific HLNs.
A $2lBN$ consist of two parts:

- The *structural part* containing the variables of the network, HLNs and LLNs, and describing the relationships between them, HLRs and LLRs. The relationships can be of two kinds:
  - HLR, represents all the labelled relationships among each HLN.
  - LLR, represents all the $is - a$ relationships among each LLN belonging to each specific HLN.
- The *probabilistic part* containing the conditional probability tables that quantify all the HLRs and LLRs introduced at the structural part.

Figure 11(b) shows an example of probabilistic part of a generic $2lBN$. We will now provide more formal definitions of the notion of two-levels Bayesian Networks.

Definition 6 ***(2lBN-structure)*** *A 2lBN is a triple $< \mathcal{N}, \mathcal{V}, \mathcal{A} >$ where*

- *$\mathcal{N}$ is the set of the HLNs each of which has a domain type τ, and corresponds to a random variable.*
- *$\mathcal{V}$ is the set of the LLNs belonging to each HLN. Each LLN corresponds to a random variable, and all the LLNs belonging to each HLN constitute the whole Bayesian Network associated to that HLN.*
- *$\mathcal{A} \subseteq \mathcal{N}^2$ is the set of directed labelled arcs between elements of $\mathcal{N}$ such that the resulting graph contains no cycles.*

Definition 7 *A 2lBN-probabilistic part related to a 2lBN-structure consist of:*

- *A LLRT, that is a Low Level Relation probability Table for each node $\in \mathcal{V}$, related to the $is - a$ relationships (LLR).*
- *A HLRT, that is a High Level Relation probability Table for each node $\in \mathcal{N}$, related to the labelled relationships (HLR).*

Definition 8 *A two-levels Bayesian Network is a pair $< \mathcal{S}, \mathcal{P} >$ where:*

- *$\mathcal{S} =< \mathcal{N}, \mathcal{V}, \mathcal{A} >$, is a 2lBN-structure*
- *$\mathcal{P}$, is the 2lBN-probabilistic part related to $\mathcal{S}$*

3.3 Ontology: Hypothesis and Requirements

As discussed in the previous chapter OWL is the Web Ontology Language recommended by W3C. The proposal of a standard language came out because all languages used to develop tools and ontologies for specific user communities (particularly in the sciences and in company-specific e-commerce applications), were not defined to be compatible with the architecture of the World Wide Web in general, and the Semantic Web in particular. OWL uses both URIs for naming and the description framework for the Web provided by RDF to add the following capabilities to ontologies:

- Ability to be distributed across many systems.
- Scalability to Web needs.
- Compatibility with Web standards for accessibility and internationalization.
- Openess and extensiblility.

An OWL ontology is an RDF graph, which is in turn a set of RDF triples. OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. ”exactly one”), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. However, the meaning of an OWL ontology is solely determined by the RDF graph. In our work we do not deal with neither data properties nor logical relations among concepts introduced by OWL. This task will be part of our immediate next steps, but a general discussion about it, is presented in chapter 5.
Intuitively, in terms of $2lBN$, an ontology is a set of relationships among knowledge of different domains. Figure 13 shows an example of ontology. We can see that it contains descriptions about four domains:

- *Company domain* describing some kind of company
- *Person domain* describing the concerps of male and female

- *Sector domain* describing the kind of sector in which a company operates
- *Project domain* describing some kind of project that can be led by a person

The ontology also contains the following relationships among those domain descriptions:

- *hasceo* connecting company domain with person domain meaning that each company can have one or more chief executive officer
- *hasSector* connecting company domain with sector domain meaning that each company operates in a specific sector
- *leads* connecting person domain with project domain meaning that each person can lead one or more projects

Those relationships are called object properties. As discussed in the previous chapter, ontologies permit to describe knowledge also by using data properties. They represent attributes of each concept of a domain. A data property can refer to both quantitative and qualitative attributes. Figure 13 shows an example of data properties related to the company domain. Each company has an initial capital, that is a quantitative attribute, and a name that is a qualitative one.
More formally, ontologies consist in general of two parts: intensional and extensional. The former part, consisting of a $TBox$, contains knowledge about concepts (i.e., classes) and complex relations between them (i.e., roles). The latter part, consisting of an $ABox$, contains knowledge about entities (i.e. individuals) and how they relate to the classes and roles from the intensional part. A general discussion about data properties is presented in chapter 5.
Now, we give our formal ontology definition.

Definition 9 *A domain concept D is a set of ontology classes and $is-a$ relationships among them, describing knowledge about a specific domain within the whole ontology.*

Note that D does not explicitly exist as a class in the ontology.

Definition 10 *A TBox ontology is a triple* $< \mathcal{D}, \mathcal{C}, \mathcal{R} >$ *where:*

- $\mathcal{D}$ *is the set of the domain concepts. Within each domain concept all the classes are in relation each other by means of* $is - a$ *relationship*
- $\mathcal{C}$ *is the set of the ontology classes.*
- $\mathcal{R}$ *is the set of object properties binding domain concepts each other.*

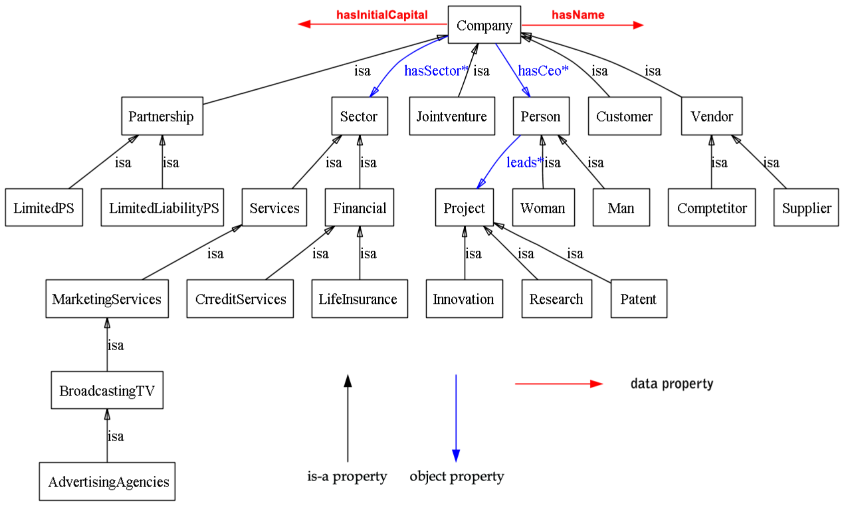


Figure 13: Example of ontology.

Figure 14 shows the $TBox$ ontology of figure 13.

Definition 11 *An* $ABox$ *ontology is a set of ontology instances* $\mathcal{I}$*. For each instance* $i \in \mathcal{I}$ $i \in \mathcal{C}$ *holds. Given a subset* $\{i\} \subseteq \mathcal{I}$*, the instances of* $\{i\}$ *can be in relation among each other by means of a subset of object properties* $\{r\} \subseteq \mathcal{R}$*, according to* $TBox$ *ontology.*

Definition 12 *An ontology* $\mathcal{O}$ *is a pair* $< \mathcal{T}, \mathcal{I} >$ *where:*

- $\mathcal{T}$ *is a triple* $< \mathcal{D}, \mathcal{C}, \mathcal{R} >$
- $\mathcal{I}$ *is the set of ontology instances*

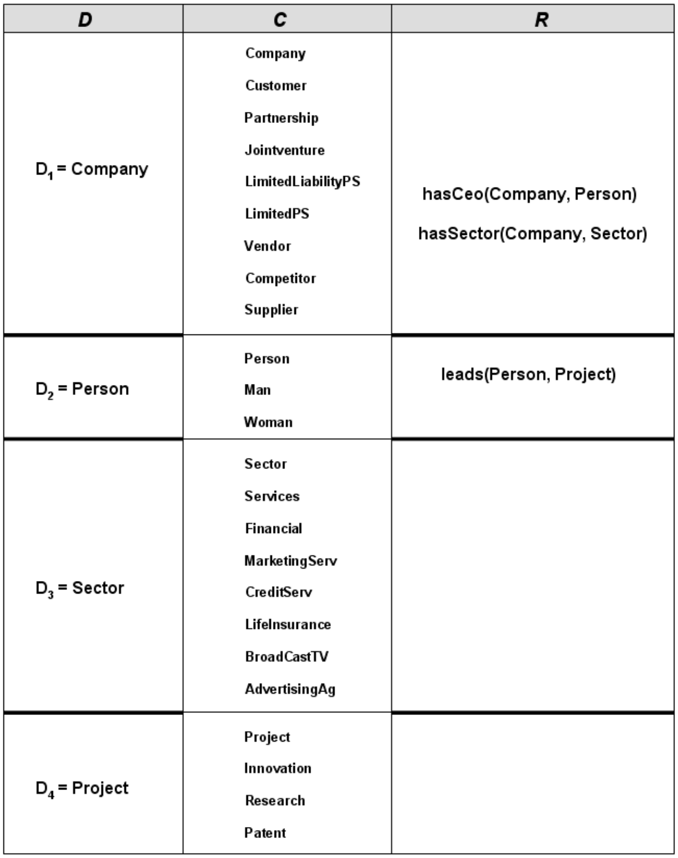
3.4 Ontology Compiling Process

In this section we define the ontology compiling process for deriving the $2lBN$ directly from the ontology. It is composed of two phases:

- Phase one: compiling $TBox$ ontology into a $2lBN$ structural part
- Phase two: compiling $ABox$ ontology into a $2lBN$ probabilistic part

Phase one maps each ontology entity to a random variable of the $2lBN$, and builds the whole structure of the Bayesian network. Phase two permits us to associate values of uncertainty to the relations modelled in the first phase. Information about the uncertainty of the classes and relations in an ontology, is here represented as probability distributions. In general they can be provided by domain experts, but in our work they are directly derived by means of the explicit knowledge stored in the ontology. We deal with two kinds of distributions. The first one, representing the probability that an arbitrary ontology instance belongs to a specific class, and the second one, representing the probability that an arbitrary ontology instance is involved in specific object properties. If class A for example, represents a concept, we treat it as a random Boolean variable of two states a and $\overline{a}$, and we interpret $P(a)$ as the prior probability that an arbitrary instance belongs to a, and $P(a|b)$ as the conditional probability that an instance of class B also belongs to class A [1]. Similarly, we can interpret $P(a), P(\overline{a}|b), P(a|\overline{b}), P(\overline{a}|\overline{b})$ with the negation interpreted as "*not belonging to*". Concerning relations among domain concepts, we treat domains and ranges of object properties as random multi-value variables, that associate instances belonging to a class to instances belonging to another class with a certain likelihood, w.r.t. the object property that they encode. In the following sections we will explain all the details of phase one and phase two.

[1]Note that the concept of "*belonging*" referred to each instance, can be thought as "*involved in* $is - a$ *relation*". $P(a)$ means that a is involved in $is - a$ relation, because each ontology instance always belongs to some class. $P(a|b)$ means that exists an $is - a$ relation between a and b.

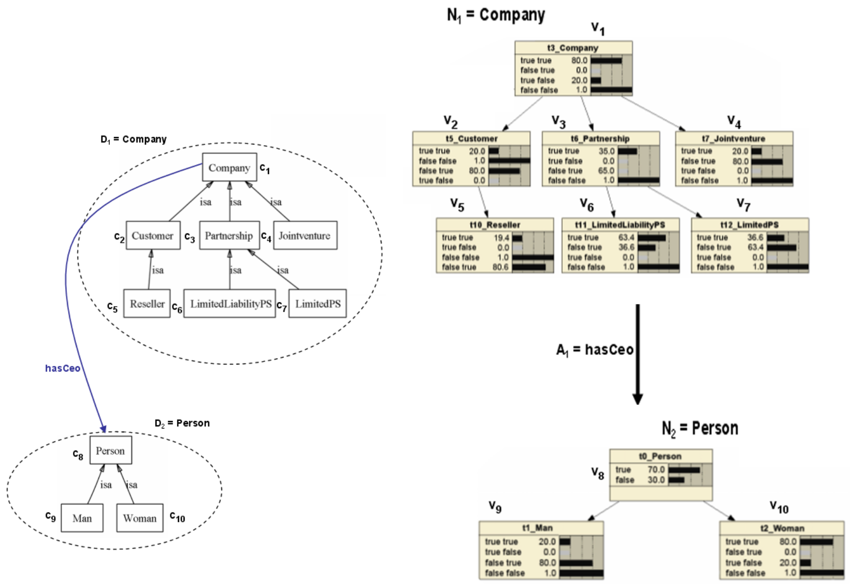


D	*C*	*R*
D_1 = Company	Company	hasCeo(Company, Person)
	Customer	hasSector(Company, Sector)
	Partnership	
	Jointventure	
	LimitedLiabilityPS	
	LimitedPS	
	Vendor	
	Competitor	
	Supplier	
D_2 = Person	Person	leads(Person, Project)
	Man	
	Woman	
D_3 = Sector	Sector	
	Services	
	Financial	
	MarketingServ	
	CreditServ	
	LifeInsurance	
	BroadCastTV	
	AdvertisingAg	
D_4 = Project	Project	
	Innovation	
	Research	
	Patent	

Figure 14: Example of $TBox$ ontology.

3.4.1 Deriving Two-level Bayesian Network Structure

In this section we define how to build the structural part of $2lBN$ starting from the $TBox$ ontology. The idea is to map each ontology class to a random Boolean variable, to find out all the domain concepts, and to map each of them to a random multi-value variable. So, at the upper level of the network each HLN represents a domain concept of the ontology, and each labelled arc represents a specific object property. At the lower level, each HLN is a Bayesian network composed of LLNs each of which represents an ontology class. Arcs of each Bayesian network encode $is-a$ ontology relation among classes. Now we introduce some definitions for



D1 = Company
Company c1
isa
c2 Customer
c3 Partnership
c4 Jointventure
c5 Reseller
c6 LimitedLiabilityPS
c7 LimitedPS
hasCeo
D2 = Person
c8 Person
c9 Man
Woman c10
N1 = Company
v1 t3_Company: true true 80.0; false true 0.0; true false 20.0; false false 1.0
v2 t5_Customer: true true 20.0; false false 1.0; false true 80.0; true false 0.0
v3 t6_Partnership: true true 35.0; true false 0.0; false true 65.0; false false 1.0
v4 t7_Jointventure: true true 20.0; false true 80.0; true false 0.0; false false 1.0
v5 t10_Reseller: true true 19.4; true false 0.0; false false 1.0; false true 80.6
v6 t11_LimitedLiabilityPS: true true 63.4; false true 36.6; true false 0.0; false false 1.0
v7 t12_LimitedPS: true true 36.6; false true 63.4; true false 0.0; false false 1.0
A1 = hasCeo
N2 = Person
v8 t0_Person: true 70.0; false 30.0
v9 t1_Man: true true 20.0; true false 0.0; false true 80.0; false false 1.0
v10 t2_Woman: true true 80.0; false true 0.0; true false 20.0; false false 1.0

Figure 15: Mapping $TBox$ ontology to $2lBN$ structure.

formalizing the ontology compiling process.
We introduce the following mapping function.

Definition 13 *Let the function Φ be*

Φ*: Ontology class* $\longrightarrow$ *booelan random variable.*

mapping each ontology class to a LLN.
Let Φ^{-1} *be the inverse function of* Φ*.*

We can observe that Φ is injective. This condition assures that the inverse function Φ^{-1} exists. It is a $partial$ function because it is not defined over the whole codomain.
Analogously we define the following functions.

Definition 14 *Let the function* Υ *be*

Υ*: Concept Domain* $\longrightarrow$ *multi-value random variable.*

mapping each ontology concept domain to a HLN.

Definition 15 *Let the function* Ψ *be*

Ψ*: Object Property* $\longrightarrow$ *Bayesian Arc.*

mapping each object property to a Bayesian arc connecting HLNs each other.

Now we can the define the ontology compiling process related to the $2lBN$ structure.

Definition 16 *The compiling process of a* $TBox$ *ontology is a function* Ω_s *such that:*

$$\begin{gathered}\Omega_s :< \mathcal{D}, \mathcal{C}, \mathcal{R} > \longrightarrow < \mathcal{N}, \mathcal{V}, \mathcal{A} >, \textit{with} \\ \Phi(\mathcal{C}) = \mathcal{V} \\ \Upsilon(\mathcal{D}) = \mathcal{N} \\ \Psi(\mathcal{R}) = \mathcal{A}\end{gathered}$$

where Φ, Υ, Ψ *are the functions defined so far.*

Figure 15 and 16 show an example of mapping of a $TBox$ ontology to a $2lBN$ structure, and an example of Ω_s function, respectively.

TBox Ontology			*2lBN*-structure - Ω_S function		
D	C	R	Υ(D)	Φ(C)	Ψ(R)
D_1 = Company	Company = c_1	hasCeo	N_1	v_1	A_1
	Customer = c_2			v_2	
	Partnership = c_3			v_3	
	Jointventure= c_4			v_4	
	Reseller= c_5			v_5	
	LimitedLiabilityPS = c_6			v_6	
	LimitedPS= c_7			v_7	
D_2 = Person	Person= c_8		N_2	v_8	
	Man = c_9			v_9	
	Woman = c_{10}			v_{10}	

Figure 16: Example of Ω_s function.

3.4.2 Deriving Two-level Bayesian Network Probabilistic Part

In this section we give a method for computing the initial probability distribution associated to 2lBN. It is very important to note that this method supposes to have very large set of instances ($ABox$), such that they form a viable sample space for probabilities. In the worst case, when instances are not available, we need some external knowledge about data, such as syntetic instances or training data sets from which we can compute the initial probability distributions, or probability tables provided by experts according to the $TBox$. Before describing that method in detail, we give the following general definition.

Definition 17 *The ontology compiling process related to the computation of the initial probability distributions is the function Ω_p such that:*

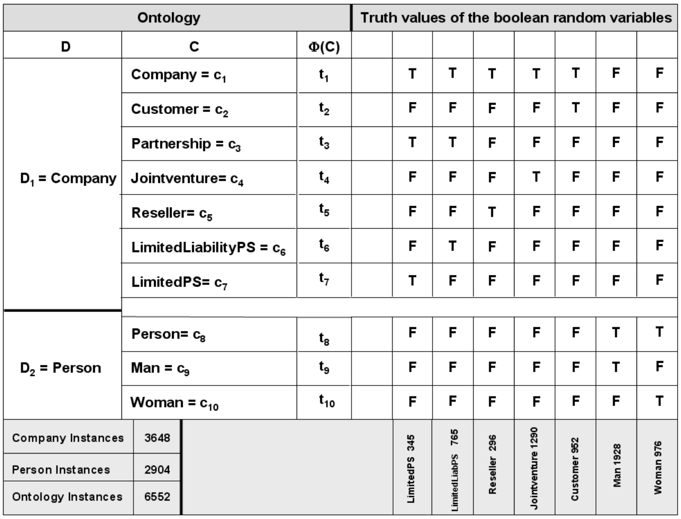
$$\Omega_p : \mathcal{I} \longrightarrow \mathcal{P}$$

where $\mathcal{I}$ is the set of all ontology instances, and $\mathcal{P}$ is the $2lBN$ probabilistic part.

Note that definition 8, in which $\mathcal{P}$ is specified, states that there exist two kinds of distributions. A low level relation probability distributions related to the $is - a$ relationships, and a high level relation probability distributions related to the labelled relationships (that are ontology object properties). In the following we discuss about both those distributions.

Low Level Relation probability distributions

This kind of distributions refers to random Boolean variables that represent ontology classes. Each domain concept D has its own distribution that describes how instances are distributed over its taxonomical description. According to section 3.4 for each class c_i, $\Phi(c_i) = v_i$ returns the associated Boolean random variable. Each of them has two truth values: v_i holds the value true when the ontology instance belongs to the class c_i, false otherwise. In practice, each domain concept D_i is a taxonomy [2],



Ontology			Truth values of the boolean random variables							
D	C	Φ(C)								
D_1 = Company	Company = c_1	t_1		T	T	T	T	T	F	F
	Customer = c_2	t_2		F	F	F	F	T	F	F
	Partnership = c_3	t_3		T	T	F	F	F	F	F
	Jointventure= c_4	t_4		F	F	F	T	F	F	F
	Reseller= c_5	t_5		F	F	T	F	F	F	F
	LimitedLiabilityPS = c_6	t_6		F	T	F	F	F	F	F
	LimitedPS= c_7	t_7		T	F	F	F	F	F	F
D_2 = Person	Person= c_8	t_8		F	F	F	F	F	T	T
	Man = c_9	t_9		F	F	F	F	F	T	F
	Woman = c_{10}	t_{10}		F	F	F	F	F	F	T
Company Instances	3648			LimitedPS 345	LimitedLiabPS 765	Reseller 296	Jointventure 1290	Customer 952	Man 1928	Woman 976
Person Instances	2904									
Ontology Instances	6552									

Figure 17: Example of instance counting.

it contains all the instances of the specific domain it represents. In other terms, D_i can be thought of as the root class of the taxonomy representing

[2]We do not consider constraints on logical relations among classes such as, intersection, disjoint, union, and so on.

that domain. Since our scenario concerns taxonomies, from this point of view all the instances not belonging to a specific class belongs to the class defined by the set difference between the taxonomy root class and that class. So, for each v_i we can define $\Phi^{-1}(\overline{v}_i)$ in terms of $\Phi^{-1}(v_i)$ as follows:

$$\Phi^{-1}(\overline{v}_i) = \Phi^{-1}(N_i) \;/\; \Phi^{-1}(v_i),$$

where / is the set difference operator between sets (or classes), and where $\Phi^{-1}(N_i) = D_i$ is the root node of the taxonomy to which $\Phi^{-1}(v_i) = c_i$ belongs. The number of instances belonging to the set difference between the ontology classes $\Phi^{-1}(N_i)$ and $\Phi^{-1}(v_i)$, is equal to the numer of instances not belonging to the class $\Phi^{-1}(v_i)$, that is equivalent to set the value of v_i to false (e.g., $\overline{v}_i$). Figure 17 shows an example of Ω_p function

Company	P(Company)
T	0.55
F	0.45

Reseller	Company	P(Reseller | Company)
T	T	0.08
F	T	0.92
T	F	0.0
F	F	1.0

Person	P(Person)
T	0.45
F	0.55

Man	Person	P(Man | Person)
T	T	0.66
F	T	0.44
T	F	0.0
F	F	1.0

Figure 18: Exampe of initial probability distribution.

counting ontology instances. It constructs a table in which the first two columns report all ontology classes and the domain concepts to which they belong. The other columns report all the possible combinations of the truth values of the random Boolean variables. Each column identifies a combination corresponding to a "state of belonging" s_i in which each instance can be in. For each of those combinations the instances belonging to that state s_i are counted. For example in figure 17, there are 3648 companies of which 345 are limited partnerships, 765 are limited liability partnerships, 296 are resellers, 1290 are jointventures, and 952 are customers. There are also observed 2904 instances of person of which 1928 are men, and 976 are women. Starting from this table, we can compute the

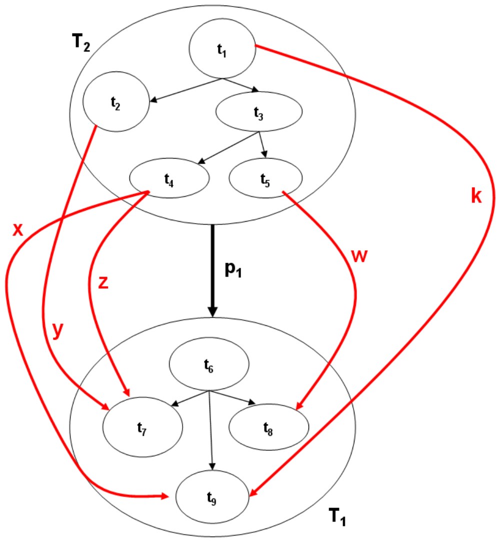


Figure 19: Conditional probability distribution of a node with a single father.

probability distribution by applying the Bayes formula in the following form:

$$P(t_i|t_j) = \frac{P(t_i \cap t_j)}{P(t_j)}$$

where $\Phi(c_i) = t_i$ and $\Phi(c_j) = t_j$.
In this context we consider that the logical relations among sets, also hold among ontology classes. In this sense we give the following definition.

Definition 18 *Let c_i, c_j two ontology classes. We have that:*

- *$c_i \cap c_j$ is the set containing the instances belonging both c_i and c_j*
- *$c_i \cup c_j$ is the set containing the instances belonging to c_i and those belonging to c_j*
- *$\bar{c}_i$is the set containing the instances not belonging to c_i*

Figure 18 shows an example of computation of both prior and conditional probability, derived by table in figure 17.

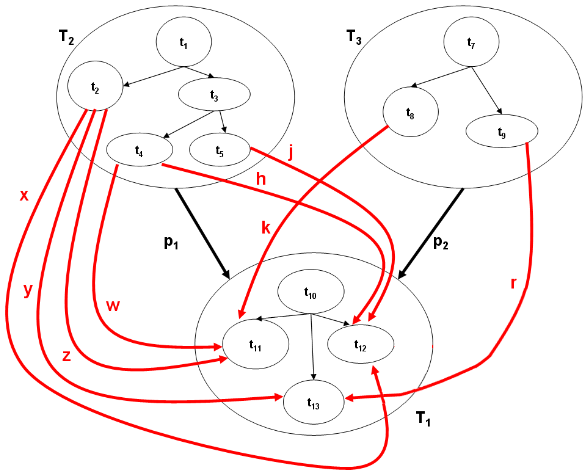


Figure 20: Computation of Conditional Probability distribution of a node with two fathers.

High Level Relations probability distribution

The second kind of distribution is related to the ontology object properties. Each domain concept is mapped to an HLN, and each of them is related to the other one via labelled arcs representing object properties. In terms of Bayesian networks, the arcs represent Bayesian conditionings among HLNs. So, we have to compute the distributions associated to each HLN w.r.t. the object properties that involve it. Each HLN corresponds to a multi-value random variable, which assumes all the possible values referring to its own domain concept value, and the values of the domain concept over which it ranges via specific labelled Bayesian conditioning. For example, the conditional probability distribution of the $hasCeo$ object property is specified by the following notation:

$$P(Person|_{hasCeo}Company)$$

$x[t_4] = x[t_9] = 139$
$y[t_2] = y[t_7] = 82$
$z[t_4] = z[t_7] = 97$
$w[t_6] = w[t_8] = 320$
$k[t_1] = k[t_9] = 81$

Total T_1 instances = 1527

$$P(T_1 = t_7 |_{p1} T_2 = t_3) = \frac{P(< t_3, p1, t_7>)}{P(< t_3, p1, \mathit{All}>)} = \frac{\frac{97}{1527}}{\frac{82 + 97 + 320}{1527}} = 0.194388$$

Figure 21: Computation of Conditional Probability distribution of a node with a single father.

This particular notation represents the probability that *hasceo* relation exists between a specific kind of company and a specific class of person, depending on how both person and company concept domains are modelled in the ontology. It is computed by applying the Bayes formula in the following form:

$$P(Person = p|_{hasCeo}Company = c) = \frac{P(<Company{=}c, hasCeo, Person{=}p>)}{P(<Company{=}c, hasCeo, Person{=}\mathcal{A}ll>)}$$

where c and p are classes belonging to person and company concept respectively, and $Person = \mathcal{A}ll$ indicates all the instances belonging to person (in this case both man and woman)[3]. Note that all the instances are counted in the HLN entered by the Bayesian conditioning arc. In the previous case, we counted all the instances of person which belongs to class p (e.g., man), and they are chief executive officers of a company that belongs to class c (e.g., supplier). Then, we counted all the instances of any person (e.g., man and woman) which is chief executive officer of a company that belongs to class c (e.g., supplier). The ratio between those quantity gives the likelihood that a man is chief executive officer of a supplier company. Figure 19 and 20 show general examples of probability distribution. In particular figure 20 shows the case in which an HLN has two income Bayesian conditionings. Figure 21 and 22 show how to compute the probability distributions of the figure 19 and 20, respectively. The

[3]The a priori probability of company HLN is trivially computed in the same way, counting the instances of company which have ceo or not, and considering all the kinds of company.

$x[t_2] = x[t_{12}] = 39$
$y[t_2] = y[t_{13}] = 32$
$z[t_2] = z[t_{11}] = 87$
$w[t_4] = w[t_{11}] = 329$
$k[t_8] = k[t_{11}] = 23$
$h[t_4] = h[t_{12}] = 15$
$j[t_5] = j[t_{12}] = 111$
$r[t_9] = r[t_{13}] =34$

Total T_1 instances = 1527

$$P(T_1 = t_{12} |_{p1,p2} T_2 = t_3, T_3 = t_9) = \frac{P(< t_3, p1, t_{12}>, < t_9, p_2, t_{12}>)}{P(< t_3, p1, \mathit{All}>, < t_9, p2, \mathit{All}>)}$$

$$= \frac{\frac{15 + 111}{1527}}{\frac{111 + 34 + 15}{1527}}$$

$$= 0.7875$$

Figure 22: Computation Conditional probaiblity distribution of a node with two fathers.

Bayes formula defined so far permits to correctly handle object properties with multiple cardinality.

Chapter 4

Inference over Bayesian Network

4.1 BQ Language

4.1.1 Introduction

The basic computation on belief networks is the computation of the belief of every node (its conditional probability) given the evidence that has been observed so far. Although evaluating Bayesian networks is, in general, NP-hard, there is a class of networks that can efficiently be solved in time linear in the number of nodes. The class is that of the polytrees. BQ language queries a bayesian network $\mathcal{R}$ satisfying the polytree properties. It is one in which the underlying undirected graph has no more than one path between any two nodes (the underlying undirected graph is the graph one gets if one simply ignores the directions on the edges). In particular, for each node t, it is possible to partition all the other nodes into two disjoint sets. The first one, is the set of nodes which are $over\ t$, meaning those nodes that are connected to t only via the fathers of t. The second one is the set of nodes which are $under\ t$, meaning those nodes that are connected to t only via the immediate descendents of t. An example of a typical polytree is shown in figure 23(a).

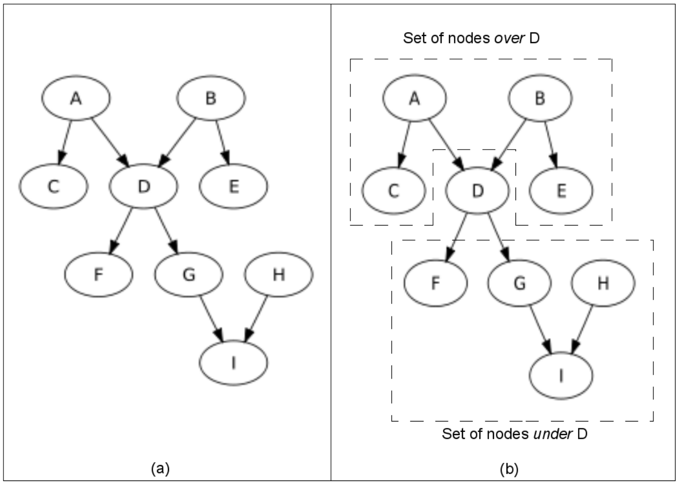


Figure 23: (a) A typical polytree. (b) under evindence and over evidence w.r.t. D node.

4.1.2 BQ Language: The syntax

According to the polytree definition, the syntax of the BQ language is defined in table 1. The general form of a query is:

$$\mathcal{P}(\text{QUERY} \mid \text{EVIDENCE}).$$

Both QUERY and EVIDENCE are polytree nodes, represented in the grammar by the syntactic cathegory $node$. Is it possible to represent both a prior probability and a conditional probability by specifying an evidence. Evidence can refer either to $is - a$ ontology relations among classes or to ontology object properties. In the latter case, the bayesian conditioning is annotated by the path composed of the arcs connecting the query node to the evidence node (e.g., the set of object properties connecting the query class to the evidence class). Since the network is a polytree, it is possible to specify the path in a unique way. Each path is represented by

the syntactic cathegory *path*. Nodes and paths refer to ontology classes and object properties respectively, according to the Φ function defined in chapter 3 section 3.1. From the physical polytree structure point of view, each evidence node is *under* or *over* a query node. An example is depicted in figure 23(b). By using our grammar, it is possible to observe evidence nodes both *under* and *over*, w.r.t. a query node. In figure 23(b), for example, D is the query node. The set of nodes *over* D is { A, B, C, E} because all those nodes are connected only to the father of D i.e., A and B. The set of nodes *under* D is { F, G, H, I} because all those nodes are connected only to the direct descendents of D i.e., F and G.
Relatively to the grammar defined in table 1, we use t_i and p_j to represent

PROBABILISTIC_QUERY ::= $\mathcal{P}$(QUERY | EVIDENCE)
QUERY ::= *node*
EVIDENCE ::= UNDER_OVER | SINGLE | ε
UNDER_OVER ::= $_{path,path}(node, node)$ | *node*,*node*
SINGLE ::= $_{path}(node)$ | *node*
node::= Ide
path::= *Rel* | *Rel.path*

Table 1: BQ grammar

a variable identifying a polytree node $\in \mathcal{R}$ (e.g., a boolean random variable), and a path binding a query node to each observed evidence node (e.g., a set of the polytree arcs among nodes), respectively.
Each path p_j can be either a simple path, that is a single polytree arc, or a composite path, that is a sequence of polytree arcs. A composite path is the concatenation of several simple paths. The concatenation operator is represented by the symbol ".". Remember that, since $\mathcal{R}$ is a polytree, each path p_j is unique in $\mathcal{R}$.
Each node t_i means that the value of the boolean variable t_i is true, meaning that we consider all the instances belonging to the class $\Phi^{-1}(t_i)$ for computing the probability distributions. Φ^{-1} is defined over t_i as stated

in definition 13. On the contrary, $\bar{t}_i$ means that the value of the boolean variable t is false, meaning that we consider all the instances not belonging to $\Phi^{-1}(t_i)$.

4.1.3 BQ Language: The operational semantic

According to the ontology definition, we distinguish two kinds of inference. The first kind is the inference over the $is - a$ relations among ontology classes, and the second one is the inference involving object porperties among ontology classes. We refer to the first inference kind by $inference\ over\ taxonomy$, and to the second one by $inference\ over\ polytree$. Implicitly, when we refer to ontology $is - a$ relationships, we deal with Low Level Nodes (LLNs); when we refer to object properties we deal with High Level nodes (HLNs).

4.2 Inference over taxonomies

We introduce some definition for specifying the BQ language operational semantic.
The following is the function needed for computing the number of instances belonging to a specific ontology class.

Definition 19 *Let* $\#_{inst}$ *be the function counting the number of the instances of each ontology class.*

So, $\#_{inst}(\Phi^{-1}(t), \mathcal{R})$ is the number of instances satisfying the condition $t = true$ in the observations table defined in the previous chapter.

Definition 20 *Let* $\#_{\overline{inst}}$ *be the function counting the number of the instances not belonging to the ontology class* $\Phi^{-1}(t_i)$.

$\#_{\overline{inst}}(\Phi^{-1}(\bar{t}_i), \mathcal{R})$ is the number of the instances satisfying the condition $t = false$ in the observations table defined in the previous chapter.

Definition 21 *Let t_1 and t_2 be two nodes in $\mathcal{R}$. $t_1 \ll t_2$ means that t_1 is under t_2 in $\mathcal{R}$ or, similarly, t_2 is over t_1 in $\mathcal{R}$, according to the polytree definition.*

Definition 22 *Let t be a node in $\mathcal{R}$. The function $root$ is defined as follows:*

$$root(t,\mathcal{R}) = \begin{cases} true & \text{if } t \text{ is the root of the taxonomy;} \\ false & \text{otherwise.} \end{cases}$$

Definition 23 *Let t be a node in $\mathcal{R}$, and $\{t_i\}$ be a set of nodes in $\mathcal{R}$. The function $father$ is defined as follows:*

$$father(t,\mathcal{R}) = \begin{cases} \oslash & \text{if } root(t,R); \\ \{t_i\} & \text{otherwise.} \end{cases}$$

where $\forall i = 1, ..., n.$ t_i is the direct ancestor of t in $\mathcal{R}$.

Notice that, since $\mathcal{R}$ is a taxonomy, in general, each node has at most one father. In this case the number of elements of $\{t_i\}$ is always equal to one. Figures 24, 25, 26, 27 contain the operational semantic rules for the BQ language. In the following we explain in detail the computation provided by each rule. Rule (1) concerns the most general probabilistic query. Given the query node t, two evidences t_1 and t_2 are obsverved, such as t_1 is *under* t, and t_2 is *over* t. We apply the Bayes formula and we obtain the following equation:

$$\mathcal{P}(t|t_1,t_2) = \frac{\mathcal{P}(t_1|t,t_2)\cdot\mathcal{P}(t|t_2)}{\mathcal{P}(t_1|t_2)}$$

We can re-write this formula as follows:

$$\mathcal{P}(t|t_1,t_2) = \mathcal{P}(t_1|t,t_2)\cdot\mathcal{P}(t|t_2)\cdot\mathcal{K}$$

where $\mathcal{K} = \frac{1}{\mathcal{P}(t_1|t_2)}$ is a normalisation factor and it is computed as shown in the following. Since t_1 and t_2 are d-separated by t, we obtain:

$$\mathcal{P}(t|t_1,t_2) = \mathcal{P}(t_1|t)\cdot\mathcal{P}(t|t_2)\cdot\mathcal{K}.$$

Since t is an evidence *over* t_1 in $\mathcal{P}(t_1|t)$, and t_2 is an evidence *over* t in $\mathcal{P}(t_1|t)$, now we can compute $\mathcal{P}(t_1|t)$ and $\mathcal{P}(t|t_2)$ separately, by applying to both rule (2). In order to compute the factor $\mathcal{K}$, we use the property that the sum of the probability of an event given an evidence, and the probability of its negation given the same evidence, is always equal to one. In our case we have:

$$\mathcal{P}(t|t_1,t_2) + \mathcal{P}(\bar{t}|t_1,t_2) = 1$$

By applying the Bayes formula to both members of the previous equation we obtain:

$$\frac{\mathcal{P}(t_1|t)\cdot\mathcal{P}(t|t_2)}{\mathcal{K}} + \frac{\mathcal{P}(t_1|\bar{t})\cdot\mathcal{P}(\bar{t}|t_2)}{\mathcal{K}} = 1$$

from which we derive that $\mathcal{K} = \mathcal{P}(t_1|t)\cdot\mathcal{P}(t|t_2)+\mathcal{P}(t_1|\bar{t})\cdot\mathcal{P}(\bar{t}|t_2)$. Replacing $\mathcal{K}$ in the first equation, we obtain:

$$\mathcal{P}(t|t_1,t_2) = \frac{\mathcal{P}(t_1|t)\cdot\mathcal{P}(t|t_2)}{\mathcal{P}(t_1|t)\cdot\mathcal{P}(t|t_2)+\mathcal{P}(t_1|\bar{t})\cdot\mathcal{P}(\bar{t}|t_2)}$$

Evidence both *under* **and** *over* **w.r.t. query** (1)

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t \\ \mathcal{P}(t_2|t,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t|t_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|\bar{t},\mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|t_1,\mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t|t_1,t_2,\mathcal{R}) \rightarrow \frac{n_1\cdot n_2}{n_1\cdot n_2+n_3\cdot n_4}}$$

Evidence *over* **w.r.t. query** (2)

$$\frac{\begin{array}{c} t_2 \gg t_1, t = \mathit{father}(t_1,\mathcal{R}) \\ \mathcal{P}(t_1|t,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t|t_2,\mathcal{R}) \rightarrow n_2 \end{array}}{\mathcal{P}(t_1|t_2,\mathcal{R}) \rightarrow n_1 \cdot n_2}$$

Evidence *under* **w.r.t. query** (3)

$$\frac{\begin{array}{c} t_2 \ll t_1 \\ \mathcal{P}(t_2|t_1,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1|\varepsilon,\mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|\bar{t}_1,\mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}_1|\varepsilon,\mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t_1|t_2,\mathcal{R}) \rightarrow \frac{n_1\cdot n_2}{n_1\cdot n_2+n_3\cdot n_4}}$$

Figure 24: Semantic rules about *under* and *over* evidence

Rule (2) is a recursive one computing the probability of the ancestors of the query node t_1 given the evidence node t_2, where the evidence is *over* the query, and it implements a $bottom - up$ reasoning schema. Since $\mathcal{R}$ is a taxonomy, each node has, at most, one father. For the first one, we have to introduce the father of the query node:

$$\mathcal{P}(t_1|t_2) = \sum_{father(t_1)} \mathcal{P}(t_1, father(t_1)|t_2)$$

The $\sum_{father(t_1)}$ notation, as specified in chapter 2, is used to indicate the fact that we have to sum all the cases of $\mathcal{P}(t_1, father(t_1)|t_2)$ among each other. The first one, when the boolean random variable $father(t_1)$ holds $true$, and the second one, when it holds $false$[1]. By using the conditional independence definition we obtain:

$$\mathcal{P}(t_1, father(t_1)|t_2) = \mathcal{P}(t_1|father(t_1), t_2) \cdot \mathcal{P}(father(t_1)|t_2)$$

Since each node is independent from all non-descendants nodes given its father, we can write:

$$\mathcal{P}(t_1|t_2) = \sum_{father(t_1)} \mathcal{P}(t_1|father(t_1)) \cdot \mathcal{P}(father(t_1)|t_2)$$

$\mathcal{P}(t_1|father(t_1))$ is computed by rules (4), (5), (6) or (7) depending on the thruth values of the nodes involved in that conditional probability, and we have to re-apply recursively rule (2) to $\mathcal{P}(father(t_1)|t_2)$. When the ancestor of t_1 coincides with the evidence node (t_2), the recursive process terminates, and we can apply rules (4), (5), (6) or (7). The process terminates also if the query node is the node root, and in this case we have to apply rules (8) or (9) depending on the truth value of that node. Rules (12) and (13) in figure 27 also deal with top-down inference with the evidence $over$ the query, but they involve the negation of the query node and the negation of the evidence node, respectively. They are used for computing the normalisation factor $\mathcal{K}$.
If t_2 becomes $under$ w.r.t. an ancestor of t_1, the recursive process stops, and we have to apply the rule (3). It computes the probability of the descendents of the query node t_1 given the evidence node t_2, where the evidence is $under$ the query, and it implements a $top - down$ reasoning schema. The basic step is the Bayes formula:

[1] In the operational semantic rules we omitted this notation for semplicity, but every time we involve a new node in a conditional probability, we have to consider all its possible values (in our case $true$ and $false$).

Positive query and positive evidence (4)

$$\frac{t = \mathit{father}(t_1, \mathcal{R}) \quad n_1 = \#_{inst}((\Phi^{-1}(t_1) \cap \Phi^{-1}(t)), \mathcal{R}), n_2 = \#_{inst}(\Phi^{-1}(t), \mathcal{R})}{\mathcal{P}(t_1|t, \mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

Negation of the evidence (5)

$$\frac{t = \mathit{father}(t_1, \mathcal{R}) \quad n_1 = \#_{inst}((\Phi^{-1}(t_1) \cap \Phi^{-1}(\bar{t})), \mathcal{R}), n_2 = \#_{inst}(\Phi^{-1}(\bar{t}), \mathcal{R})}{\mathcal{P}(t_1|\bar{t}, \mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

Negation of the query (6)

$$\frac{t = \mathit{father}(t_1, \mathcal{R}) \quad n_1 = \#_{inst}((\Phi^{-1}(\bar{t}_1) \cap \Phi^{-1}(t)), \mathcal{R}), n_2 = \#_{inst}(\Phi^{-1}(t), \mathcal{R})}{\mathcal{P}(\bar{t}_1|t, \mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

Negation of both query and evidence (7)

$$\frac{t = \mathit{father}(t_1, \mathcal{R}) \quad n_1 = \#_{inst}((\Phi^{-1}(\bar{t}_1) \cap \Phi^{-1}(\bar{t})), \mathcal{R}), n_2 = \#_{inst}(\Phi^{-1}(\bar{t}), \mathcal{R})}{\mathcal{P}(\bar{t}_1|\bar{t}, \mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

Figure 25: Rules about the direct computation of the conditional probability

Positive root node (8)

$$\frac{root(t, \mathcal{R}) \qquad n = \frac{\#_{inst}(\Phi^{-1}(t), \mathcal{R})}{\#(\mathcal{A}ll, \mathcal{R})}}{\mathcal{P}(t|\varepsilon, \mathcal{R}) \rightarrow n}$$

Negation of the root node (9)

$$\frac{root(t, \mathcal{R}) \qquad n = \frac{\#_{inst}(\Phi^{-1}(\overline{t}), \mathcal{R})}{\#(\mathcal{A}ll, \mathcal{R})}}{\mathcal{P}(\overline{t}|\varepsilon, \mathcal{R}) \rightarrow n}$$

Positive no root node (10)

$$\frac{\overline{root}(t, \mathcal{R}), t_1 = father(t, \mathcal{R}) \qquad \mathcal{P}(t|t_1, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1|\varepsilon, \mathcal{R}) \rightarrow n_2}{\mathcal{P}(t|\varepsilon, \mathcal{R}) \rightarrow n_1 \cdot n_2}$$

Negation of the no root node (11)

$$\frac{\overline{root}(t, \mathcal{R}), t_1 = father(t, \mathcal{R}) \qquad \mathcal{P}(\overline{t}|t_1, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1|\varepsilon, \mathcal{R}) \rightarrow n_2}{\mathcal{P}(\overline{t}|\varepsilon, \mathcal{R}) \rightarrow n_1 \cdot n_2}$$

Figure 26: Rules about the direct computation of the prior probability

$$\mathcal{P}(t_1|t_2) = \frac{\mathcal{P}(t_2|t_1)\cdot\mathcal{P}(t_1)}{\mathcal{P}(t_2)} = \mathcal{P}(t_2|t_1)\cdot\mathcal{P}(t_1)\cdot\mathcal{K},$$

where $\mathcal{K} = \frac{1}{\mathcal{P}(t_2)}$ is a normalisation factor. It is very important to note that this rule transfoms a bottom-up inference reasoning schema ($\mathcal{P}(t_1|t_2)$) into a top-down inference reasoning schema ($\mathcal{P}(t_2|t_1)$) except for $\mathcal{K}$ and $\mathcal{P}(t_1)$. So, on $\mathcal{P}(t_2|t_1)$ we can apply the recursive rule (2), and then we can compute the $\mathcal{K}$ factor in the same way shown before:

$$\mathcal{P}(t_1|t_2) + \mathcal{P}(\bar{t}_1|t_2) = 1$$

Applying the Bayes formula to both members of the previous equation we obtain:

$$\frac{\mathcal{P}(t_2|t_1)\cdot\mathcal{P}(t_1)}{\mathcal{K}} + \frac{\mathcal{P}(t_2|\bar{t}_1)\cdot\mathcal{P}(\bar{t}_1)}{\mathcal{K}} = 1$$

from which we derive that $\mathcal{K} = \mathcal{P}(t_2|t_1)\cdot\mathcal{P}(t_1)+\mathcal{P}(t_2|\bar{t}_1)\cdot\mathcal{P}(\bar{t}_1)$. Replacing $\mathcal{K}$ in the first equation we introduced, we obtain:

$$\mathcal{P}(t_1|t_2) = \frac{\mathcal{P}(t_2|t_1)\cdot\mathcal{P}(t_1)}{\mathcal{P}(t_2|t_1)\cdot\mathcal{P}(t_1)+\mathcal{P}(t_2|\bar{t}_1)\cdot\mathcal{P}(\bar{t}_1)}$$

$\mathcal{P}(t_1)$ is trivially computable. In fact, from the physical network structure point of view, we know how we can reach the evidence node (that is *under* the query node) passing via the query node, and we know how we can reach the query node from each node which is *over* the query node. Futhermore, each of this paths is unique because our network is a polytree. So, in order to know the prior probability of the query node, we have to consider all the possible paths connecting each node *over* the query node to the query node itself. In the case of taxonomies each node has, at most, one father, and the computation of the prior probability is trivial as shows rule (10). We involve recursively the father of the query node until the query node becomes the root node. The basic step is the following:

$$\mathcal{P}(t_1) = \sum_{father(t_1)} \mathcal{P}(t_1|father(t_1)\cdot\mathcal{P}(father(t_1))$$

Rule (11) is equal to (10), but it computes the prior probability of the negation of a query node. When the recursive processes of (10) and (11) terminate, we can apply the rules (8) and (9) over the root node, respectively.

Normalisation Factor Computation (12)

$$\frac{t_2 \gg t_1, t = \mathit{father}(t_1, \mathcal{R}) \quad \mathcal{P}(\bar{t}_1|t, \mathcal{R}) \to n_1, \mathcal{P}(t|t_2, \mathcal{R}) \to n_2}{\mathcal{P}(t_1|t_2, \mathcal{R}) \to n_1 \cdot n_2}$$

Normalisation Factor Computation (13)

$$\frac{t_2 \gg t_1, t = \mathit{father}(t_1, \mathcal{R}) \quad \mathcal{P}(t_1|t, \mathcal{R}) \to n_1, \mathcal{P}(t|\bar{t}_2, \mathcal{R}) \to n_2}{\mathcal{P}(t_1|\bar{t}_2, \mathcal{R}) \to n_1 \cdot n_2}$$

Figure 27: Semantic rules about the normalisation factor computing

Rules (4), (5), (6), (7), (8) and (9) are defined according to the method for computing the initial probability distribution described in the previous chapter. In particular, rules (4), (5), (6), (7) compute the conditional probability distribution of each node t given its father $t_1 = father(t)$ considering all the possible truth values of the query and the evidence. In Rule (5), for example, the evidence is the negation of the father of the query. Since we deal with taxonomies, $\Phi^{-1}(t_1) \cap \Phi^{-1}(\bar{t})), \mathcal{R}) = \oslash$ holds, and $\mathcal{P}(t|\bar{t}_1) = 0$ for each taxonomy. But, when we will deal with taxonomies with multiple inheritance, this condition will not hold anymore.

4.2.1 An Example of Inference over taxonomy

The following is a simple example of inference over taxonomies by using the rules of the semantics defined so far. Figure 29 shows a fragment of the taxonomy related to Legal Entities taken from the ontology repository of the European MUSING Project (Sol). All the ontology fragments presented in this thesis, are taken from that repository. According to the method defined in the previous chapter, the taxonomy is compiled

$$\dfrac{\dfrac{\begin{array}{c} n_4 = \#_{inst}((\varPhi^{-1}(t_4),\mathcal{R}) \\ n_3 = \#_{inst}((\varPhi^{-1}(t_{13}),\mathcal{R}) \end{array}}{\mathcal{P}(t_{13}|t_9,\mathcal{R}) \rightarrow n_1 = \frac{n_3}{n_4}} \qquad \dfrac{\dfrac{\begin{array}{c} n_8 = \#_{inst}((\varPhi^{-1}(t_4),\mathcal{R}) \\ n_7 = \#_{inst}((\varPhi^{-1}(t_9),\mathcal{R}) \end{array}}{\mathcal{P}(t_9|t_4,\mathcal{R}) \rightarrow n_5 = \frac{n_7}{n_8}} \quad \dfrac{\begin{array}{c} n_{10} = \#_{inst}((\varPhi^{-1}(t_3),\mathcal{R}) \\ n_9 = \#_{inst}((\varPhi^{-1}(t_4),\mathcal{R}) \end{array}}{\mathcal{P}(t_4|t_3,\mathcal{R}) \rightarrow n_6 = \frac{n_9}{n_{10}}}}{\mathcal{P}(t_9|t_3,\mathcal{R}) \rightarrow n_2 = n_5 \cdot n_6}}{\mathcal{P}(t_{13}|t_3,\mathcal{R}) \rightarrow n_1 \cdot n_2}$$

Figure 28: Example of semantic rules derivation.

into a bayesian network satisfying the polytree properties. Then, starting from its instances, all the initial distributions are computed. The result is shown in figure 30. Function Φ maps each taxonomy class to a t_i bayesian node in the following way:

$$\begin{array}{llll}
\Phi(LegalEntity) & = t_1 & \Phi(Person) & = t_2 \\
\Phi(Company) & = t_3 & \Phi(Vendor) & = t_4 \\
\Phi(Customer) & = t_5 & \Phi(Partnership) & = t_6 \\
\Phi(JointVenture) & = t_7 & \Phi(Competitor) & = t_8 \\
\Phi(Supplier) & = t_9 & \Phi(Reseller) & = t_{10} \\
\Phi(LimitedLiability) & = t_{11} & \Phi(Limited) & = t_{12} \\
\Phi(PCSupplier) & = t_{13} & \Phi(OtherSupplier) & = t_{14}
\end{array}$$

For example, if we want to know which is the likelyhood of a legal entity to be a commercial personal computer selling company, given that it is a company, we have to express this query as follows:

$$\mathcal{P}(PCSupplier|Company)$$

What we know, that is the evidence, is that our observations are legal entities that must be observed only in the company sub-space (and not, for example, in the person sub-space). Then, what we want to know is

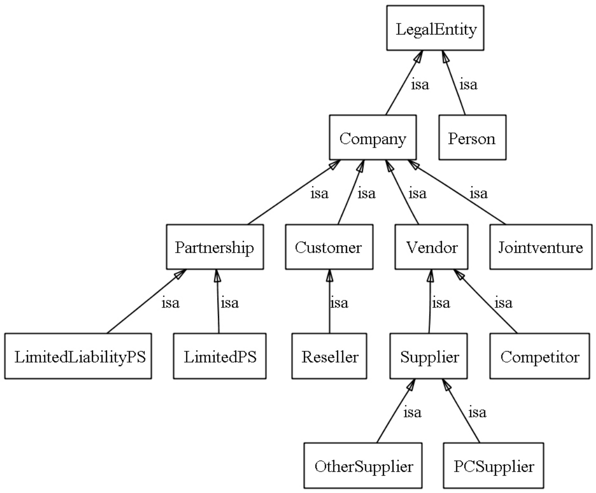


Figure 29: Fragment of the Legal Entity taxonomy of the Musing project.

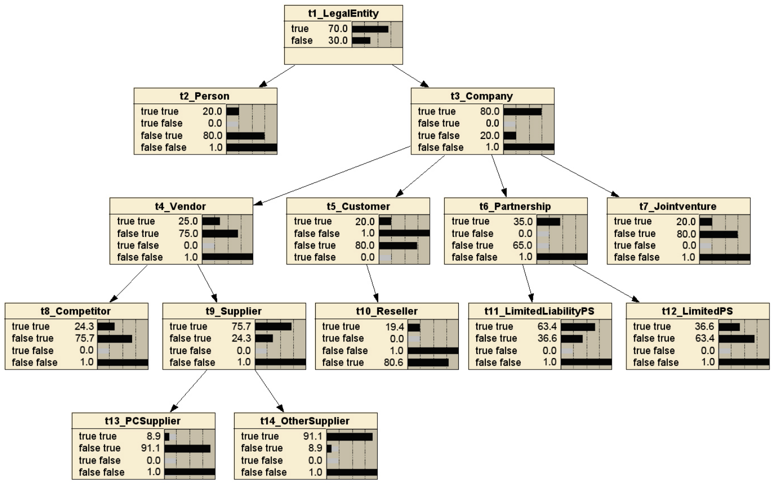


Figure 30: Legal Entity taxonomy compiled into the bayesian network

about entities which are personal computer sellers. Figure 28 shows how those rules work on the example reported in figure 30. The conditional probability we have to compute is $\mathcal{P}(t_{13}|t_3, \mathcal{R})$. The candidate rules to be applied are (2) and (3) in figure 24, and (4) in figure 25. Rule (4) can not be applied because the condition on the father of t_{13} doesn't hold. Concerning rules (2) and (3), the evidence t_3 is *over* the query t_{13}, the condition $t_3 \gg t_{13}$ holds, and we can apply rule (2). It decomposes the original probability into the product of two other probabilities:

$$\mathcal{P}(t_{13}|t_3, \mathcal{R}) = \mathcal{P}(t_{13}|t_9, \mathcal{R}) \cdot \mathcal{P}(t_9|t_3, \mathcal{R}).$$

Since the evidence is over the query, the method computes the probability of the query (t_{13}) given its father (t_9), and it is re-applied to the new quey node (t_9). Concerning the first factor, the condition $t_9 = father(t_{13}, \mathcal{R})$ holds and we can apply rule (4). It computes the ratio between PC supplier entities and supplier entities, and the process terminates. Rule (2) is recursively applied to the second factor, and it decomposes the factor into two other probabilities:

$$\mathcal{P}(t_9|t_3, \mathcal{R}) = \mathcal{P}(t_9|t_4, \mathcal{R}) \cdot \mathcal{P}(t_4|t_3, \mathcal{R}).$$

On both the factors, the condition stating that the evidence node is the father of the query node holds, and the rule (4) is applied. In the first case, it is computed the ratio between supplier entities and vendor entities, and in the second one, it is computed the ratio between vendor entities and company entities.

4.3 Inference over taxonomies with multiple inheritance

Typically, ontologies can define taxonomies with multiple inheritance. Each taxonomy class can be a subclass of two or many calsses. From the bayesian network point of you, it means that each node can have one or many fathers. We can modify our rules for dealing with this kind of taxonomy. Figure 31 shows an example of taxonomy with multiple inheritance, in fact Chairman class is a subclass of both Officer and Authority.

In order to make inference over this kind of network, we have first to modify rule (2) and (3) for involving all the fathers of each node along the inference path. In table 32 the new rules are presented. We need to

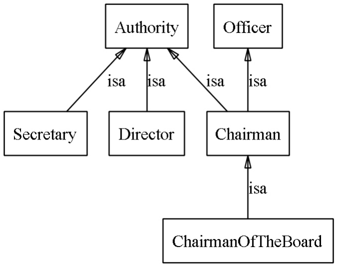


Figure 31: An example of taxonomy with multiple inheritance.

introduce the following definitions.

Definition 24 ***(Path Uniqueness Property)****. A sequence of n arcs belonging to a polytree $\mathcal{R}$ is called path of $\mathcal{R}$, with $n \geq 1$. In $\mathcal{R}$ each node can be reached by each other node via an unique path. We call this feature as path uniqueness property.*

Definition 25 *Let t_1 and t_2 be nodes in $\mathcal{R}$, such as $t_2 \ll t_1$. $desc(t_2, t_1, \mathcal{R})$ is the function returning the node t, if it exists, which is both descendent of t_1 and father of t_2.*

Definition 26 *Let t_1, t_2 be nodes in $\mathcal{R}$. The function $ances$ is defined as follows:*

$$ances(t_1, t_2, \mathcal{R}) = \begin{cases} true & \textit{if a path from } t_1 \textit{ to } t_2 \textit{ exists, and } t_1 \ll t_2 \\ false & \textit{otherwise.} \end{cases}$$

Figure 33 shows an example of *ances* function. Given the node evidence $t_2 = t_\varepsilon$, the only father of t_q, starting from which a path connecting it to t_ε exists, is t_{13} as figure 33 shows. Note that $t_{13} \ll t_\varepsilon$.
In the following we use $\mathcal{C}$ to indicate the function counting the number of nodes of a set of nodes $\{\mathrm{t}\}$. Concerning evidence *over* query, the new rule $(2bis)$, is still implemented by a recursive $bottom - up$ inference schema,

Evidence *over* w.r.t. query $(2bis)$

$$\frac{\begin{array}{c} t_2 \gg t_1, \{t\} = \mathit{father}(t_1, \mathcal{R}), t_2 not \in \{t\} \\ \mathcal{P}(t_1|\{t\}, \mathcal{R}) \rightarrow n_1, \exists i.t_i \in \{t\} : \mathit{ances}(t_2, t_i, \mathcal{R}), \mathcal{P}(t_i|t_2, \mathcal{R}) \rightarrow n_2 \\ \forall j : ((t_j \in \{t\}) \wedge \overline{\mathit{ances}}(t_2, t_j, \mathcal{R})).\mathcal{P}(t_j|\varepsilon, \mathcal{R}) \rightarrow n_j \end{array}}{\mathcal{P}(t_1|t_2, \mathcal{R}) \rightarrow n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t\})-1)} n_k}$$

Evidence *under* w.r.t. query $(3bis)$

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists k.t_k = \mathit{desc}(t_2, t_1) \\ \mathcal{P}(t_2|t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|t_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|\overline{t}_1, \mathcal{R}) \rightarrow n_4, \\ \mathcal{P}(t_1|\varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\overline{t}_1|\varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(t_1|t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{n_1 \cdot n_2 \cdot n_3 + n_1 \cdot n_4 \cdot n_5}}$$

Evidence *under* w.r.t. query (query is a father of evidence) (3.1)

$$\frac{\begin{array}{c} t_2 \ll t_1, t_1 = \mathit{father}(t_2) \\ \mathcal{P}(t_2|t_1, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|\overline{t}_1, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\overline{t}_1|\varepsilon, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t_1|t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

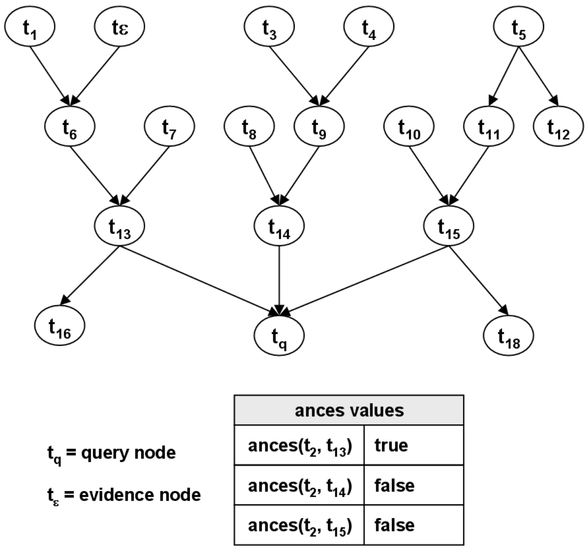
Probability distribution of a node given its $\mathcal{C}(\{t\})$ fathers (14)

$$\frac{\begin{array}{c} n_1 = \#_{inst}((\Phi^{-1}(t) \cap (\bigcap_{i=1}^{\mathcal{C}(\{t\})} \Phi^{-1}(t_i))), \mathcal{R}) \\ n_2 = \#_{inst}(\bigcap_{i=1}^{\mathcal{C}(\{t\})} \Phi^{-1}(t_i), \mathcal{R}) \end{array}}{\mathcal{P}(t|\{t\}, \mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

Evidence is one of the fathers of the query (15)

$$\frac{\begin{array}{c} \{t\} = \mathit{father}(t_1, \mathcal{R}), t_2 \in \{t\}, \mathcal{C}(\{t\}) \geq 2 \\ \forall i.t_i \in \{\{t\}/t_2\} : \mathcal{P}(t_i|\varepsilon, \mathcal{R}) \rightarrow n_i \\ \mathcal{P}(t_1|\{t\}, \mathcal{R}) \rightarrow n_1 \end{array}}{\mathcal{P}(t_1|t_2, \mathcal{R}) \rightarrow n_1 \cdot \prod_{i=1}^{(\mathcal{C}(\{t\})-1)} n_i}$$

Figure 32: Semantic rules about taxonomy evidence with multiply inheritance



ances values	
ances(t_2, t_{13})	true
ances(t_2, t_{14})	false
ances(t_2, t_{15})	false

Figure 33: Example of $ances$ function.

computing the probability of each ancestor of the query, given the evidence, until either an ancestor reaches the evidence or the evidence becomes $under$ the query. W.r.t. rule (2), we have to introduce all the fathers of the query node t_q:

$$\mathcal{P}(t_q|t_2) = \sum_{father(t_q)} \mathcal{P}(t_q, father(t_q)|t_2) = \sum_{\{t_q\}} \mathcal{P}(t_q, \{t_q\}|t_2)$$

where $\{t_q\} = father(t_q)$ is the set of fathers of t_q ($\mathcal{C}(\{t_q\})$). By using the conditional independence definition we obtain:

$$\mathcal{P}(t_q, \{t_q\}|t_2) = \mathcal{P}(t_q|\{t_q\}, t_2) \cdot \mathcal{P}(\{t_q\}|t_2)$$

Since each node is independent from all non-descendants nodes given all its fathers, we can write:

$$\mathcal{P}(t_q|t_2) = \sum_{\{t_q\}} \mathcal{P}(t_q|\{t_q\}) \cdot \mathcal{P}(\{t_q\}|t_2)$$

Since all the fathers of t_q are d-separated, we can re-write the previous equation as:

$$\mathcal{P}(t_q|t_2) = \sum_{\{t_q\}} \mathcal{P}(t_q|\{t_q\}) \cdot \mathcal{P}(t_k|t_2) \cdot \mathcal{P}(t_{k+1}|t_2) \cdot \cdots \cdot \mathcal{P}(t_{k+n}|t_2)$$

where $t_k, ..., t_{k+n} \in \{t_q\}$. We note that the terms of the previous equation are:

(a) The probability of the query node t_q given all its fathers.

(b) The probabilities of each father of t_q given the evidence.

Concerning (a), $\mathcal{P}(t_q|\{t_q\})$ factor is computed by applying rule (14) [2]. According to what explained in chapter 3, it computes the probability of a node given all its fathers, applying the Bayes formula. It is equal to the number of ontology instances belonging to the intersection of all the father classes and the query class, on the number of ontology instances belonging to the intersection of all the father classes.
Regarding (b) we can observe that d-separation property permits us to discard the evidence over each father node, when we compute the probabilities of the other father nodes. So, we select that father t_i of t_q for which $ances(t_q, t_i)$ holds. All the other fathers t_j are independent from the evidence, because each father node is d-separated from each other and the evidence is only over t_i. This result derives directly from the polytree structure. So, we obtain:

$$\mathcal{P}(t_q|t_2) = \sum_{\{t_q\}} \mathcal{P}(t_q|\{t_q\}) \cdot \mathcal{P}(t_i|t_2) \cdot \prod_{j=1}^{(\mathcal{C}(\{t_q\})-1)} \mathcal{P}(t_j)$$

where $ances(t_q, t_i) \wedge t_i \in \{t_q\}$, and $j \neq i \wedge t_j \in \{t_q\}$, as rule $(2bis)$ shows. All $\mathcal{P}(t_j)$ factors can be computed applying rules (8), (9), (10), (11) in figure 26, depending on if t_j is a root node or not.
Concerning the $\mathcal{P}(t_i|t_2)$ factor, if t_i has only t_2 as father, it is computed by rules (4), (5), (6), (7) in figure 25, depending on the truth values of t_i and t_2.

[2]In practice, we have to sum all the probabilities of the cases of $\mathcal{P}(t_q|\{t_q\})$, on the basis of the truth values of all $\{t_q\}$ nodes. For example, if t_q has two fathers t_1 and t_2, we have to compute $\sum_{\{t_1,t_2\}} \mathcal{P}(t_q|t_1, t_2)$ as the sum of four cases of $\mathcal{P}(t_q|t_1, t_2)$; the first one in which we substitute $\bar{t}_1$ to t_1, the second one in which we substitute $\bar{t}_2$ to t_2, the third one in which we substitute both $\bar{t}_1$ and $\bar{t}_2$ to t_1 and t_2 respectively, and the last one that is the original version. For semplicity, this kind of computations are omitted in the operational semantic rules. Each rule computing each of all the different cases of the conditional probability, is equal to rule (14) except for the truth values of the father nodes.

If t_i has two or more fathers, and the evidence t_2 is one of its fathers, we have to apply rule (15). In this case we have to involve in $\mathcal{P}(t_i|t_2)$, all the other fathers of t_i, $\{t_i\} = father(t_i)$:

$$\mathcal{P}(t_i|t_2) = \sum_{t_j \in \{t_i\}}^{t_j \neq t_2} \mathcal{P}(t_i|\{t_i\}) \cdot \mathcal{P}(t_2|\{\{t_i\}/t_2\})$$

where $\{\{t_i\}/t_2\}$ indicates the set of fathers of t_i excluding t_2. Since each father is d-separated from the others, we obtain:

$$\mathcal{P}(t_i|t_2) = \sum_{t_j \in \{t_i\}}^{t_j \neq t_2} \mathcal{P}(t_i|t_j) \cdot \prod_{t_k \in \{\{t_i\}/t_2\}} \mathcal{P}(t_k),$$

as rule (15) shows in table 32.
Rule $(3bis)$ is very similar to rule (3) but, since each node can have many fathers, we have to choose the father of t_2 which is a descendent of t_1. The function $desc(t_2, t_1)$ selects that node, called t_{desc}. By applying the independence property we have that $\mathcal{P}(t_2, t_{desc}|t_1) = \mathcal{P}(t_2|t_1, t_{desc}) \cdot \mathcal{P}(t_{desc}|t_1)$, and we can write:

$$\mathcal{P}(t_2|t_1) = \sum_{t_{desc}} \mathcal{P}(t_2|t_{desc}, t_1) \cdot \mathcal{P}(t_{desc}|t_1)$$

Now, an evidence node is always d-separated from a query node, w.r.t. each node satifying the property of t_{desc}. In this way we can simplify the previous equation as the following:

$$\mathcal{P}(t_2|t_1) = \sum_{t_{desc}} \mathcal{P}(t_2|t_{desc}) \cdot \mathcal{P}(t_{desc}|t_1)$$

The two factors in which $\mathcal{P}(t_2|t_1)$ is decomposed are n_1 and n_2 in rule $(3bis)$, respectively. n_3 is the a priori probability of the query, deriving from the Bayes formula, and n_4, n_5 are deriving directly from the computation of the normalisation factor, as described in section 4.2.
Rule (3.1) is very similar to $(3bis)$, but it deals with the case in which the evidence is a direct descendent of query. This is the situation in which is required only to apply the Bayes formula, in order to change the evidence from *under* to *over* w.r.t. query. Then, on that factor we have to apply rule (15). The computation of the other resulting factor, that are $\mathcal{K}$ and the a priori probability, is identical to the previous one.

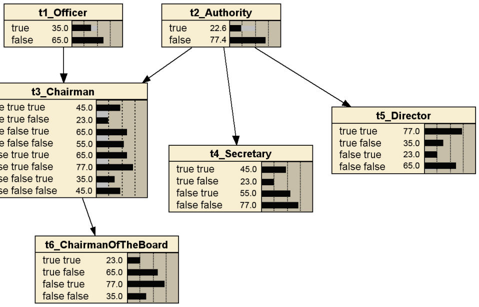


Figure 34: The taxonomy with multiple inheritance compiled into the Bayesian Network.

4.3.1 An Example of Inference over taxonomy with multiple inheritance

The following is a simple example of inference over taxonomies by using the rules of the semantics defined so far. Figure 31 shows a fragment of a taxonomy related to job positions. As you can see it is not a classical taxonomy, in fact $Authority$ and $Officer$ are root classes. $Chairman$ class is subclass of both $Authority$ and $Officer$ class. Now, we show how it is possible to make inference over this kind of taxonomy by using our new rules. The related bayesian network is depicted in figure 34.
Suppose we want to answer to the following query:

Which is the probability of an entity to be an Authority, given that it is a Chairman of the board ?

By using our syntax we can express the previous query in the following way:

$$P(Authority|ChairmanOfTheBorad) \equiv P(t_2|t_6)$$

In figure 35 is reported an example of derivation of the semantic rules for solving the previous bayesian query.
Since the evidence is a single node, the rules which are candidate to be applied to $P(t_2|t_6)$, are rules $(2bis)$, $(3bis)$ and (3.1). W.r.t. the last ones,

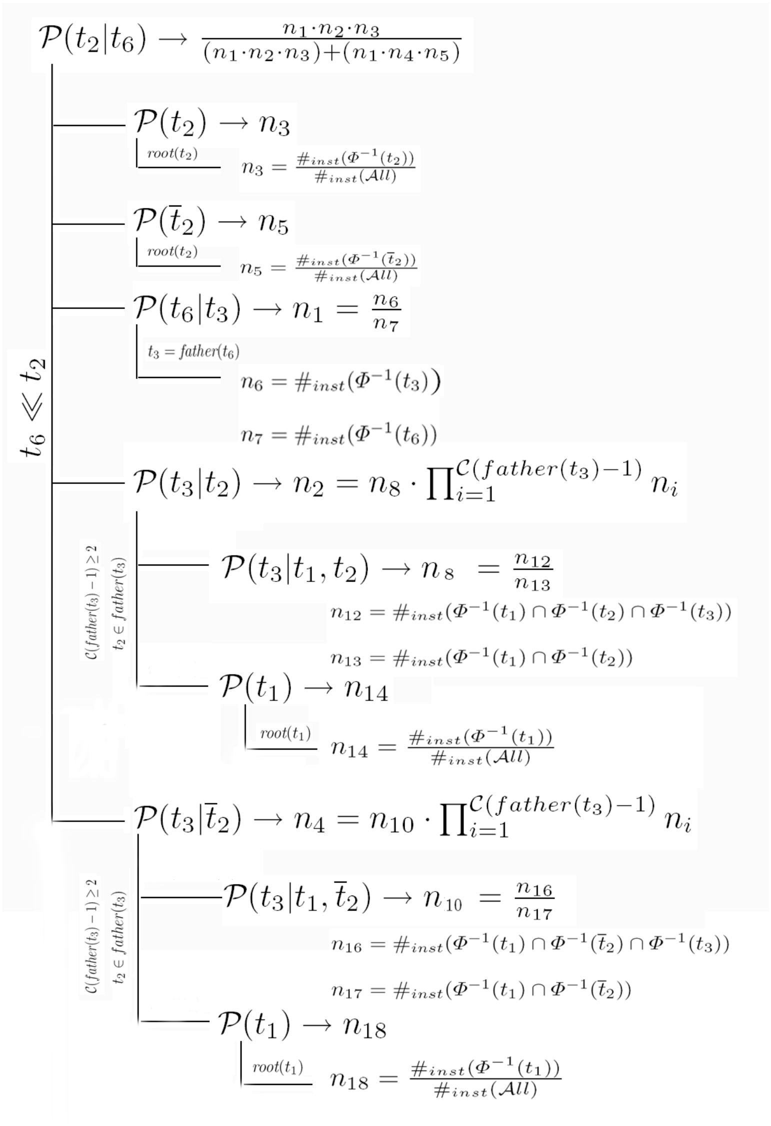

$\mathcal{P}(t_2|t_6) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{(n_1 \cdot n_2 \cdot n_3)+(n_1 \cdot n_4 \cdot n_5)}$
$t_6 \lll t_2$
$\mathcal{P}(t_2) \rightarrow n_3$
$root(t_2)$
$n_3 = \frac{\#_{inst}(\Phi^{-1}(t_2))}{\#_{inst}(\mathcal{A}ll)}$
$\mathcal{P}(\overline{t}_2) \rightarrow n_5$
$root(t_2)$
$n_5 = \frac{\#_{inst}(\Phi^{-1}(\overline{t}_2))}{\#_{inst}(\mathcal{A}ll)}$
$\mathcal{P}(t_6|t_3) \rightarrow n_1 = \frac{n_6}{n_7}$
$t_3 = father(t_6)$
$n_6 = \#_{inst}(\Phi^{-1}(t_3))$
$n_7 = \#_{inst}(\Phi^{-1}(t_6))$
$\mathcal{P}(t_3|t_2) \rightarrow n_2 = n_8 \cdot \prod_{i=1}^{\mathcal{C}(father(t_3)-1)} n_i$
$\mathcal{C}(father(t_3)) - 1) \geq 2$
$t_2 \in father(t_3)$
$\mathcal{P}(t_3|t_1, t_2) \rightarrow n_8 = \frac{n_{12}}{n_{13}}$
$n_{12} = \#_{inst}(\Phi^{-1}(t_1) \cap \Phi^{-1}(t_2) \cap \Phi^{-1}(t_3))$
$n_{13} = \#_{inst}(\Phi^{-1}(t_1) \cap \Phi^{-1}(t_2))$
$\mathcal{P}(t_1) \rightarrow n_{14}$
$root(t_1)$
$n_{14} = \frac{\#_{inst}(\Phi^{-1}(t_1))}{\#_{inst}(\mathcal{A}ll)}$
$\mathcal{P}(t_3|\overline{t}_2) \rightarrow n_4 = n_{10} \cdot \prod_{i=1}^{\mathcal{C}(father(t_3)-1)} n_i$
$\mathcal{C}(father(t_3)) - 1) \geq 2$
$t_2 \in father(t_3)$
$\mathcal{P}(t_3|t_1, \overline{t}_2) \rightarrow n_{10} = \frac{n_{16}}{n_{17}}$
$n_{16} = \#_{inst}(\Phi^{-1}(t_1) \cap \Phi^{-1}(\overline{t}_2) \cap \Phi^{-1}(t_3))$
$n_{17} = \#_{inst}(\Phi^{-1}(t_1) \cap \Phi^{-1}(\overline{t}_2))$
$\mathcal{P}(t_1) \rightarrow n_{18}$
$root(t_1)$
$n_{18} = \frac{\#_{inst}(\Phi^{-1}(t_1))}{\#_{inst}(\mathcal{A}ll)}$

Figure 35: Derivation rules for computing $P(t_2|t_6)$.

the condition $t_6 \ll t_2$ holds, but the evidence is not a direct discendent of the query, so we have to apply rule $(3bis)$, implementing the Bayes formula. It reverses t_6 with t_2 and it makes inference over $P(t_6|t_2)$:

$$P(t_2|t_6) = P(t_6|t_2) \cdot P(t_2) \cdot \mathcal{K}, \tag{4.1}$$

where $\mathcal{K} = \frac{1}{P(t_6)}$. The rule checks for that node satisfying the following condition:

$$\exists k.t_k = desc(t_2, t_6).$$

t_3 is the node satisfying this condition. As you can see in figure 34, t_3 is the father of t_6 which is descendent of t_2. So we have that:

$$P(t_6|t_2) = P(t_6|t_3) \cdot P(t_3|t_2). \tag{4.2}$$

On $P(t_6|t_3)$ we can apply rule (4) because t_3 is the unique father of t_6. It computes this factor by counting opportunely the ontology instances of $\Phi^{-1}(t_6)$ and $\Phi^{-1}(t_3)$. Futhermore, $P(t_3|t_2)$ satisfies the conditions of rule (15):

$$\{t_2, t_1\} = \mathit{father}(t_3, \mathcal{R}), t_2 \in \{t_2, t_1\}, \mathcal{C}(\{t_2, t_1\}) \geq 2.$$

Rule (15) involves t_1 node, that is the other father of t_3. As you can see in figure 35, $(\mathcal{C}(\mathit{father}(t_3)) - 1)$ is equal to 1, so $\prod_{i=1}^{1} n_i$ has only one term n that is equal to $P(t_1)$. The rule decomposes $P(t_3|t_2)$ into the following way:

$$P(t_3|t_2) = P(t_3|t_1, t_2) \cdot P(t_1). \tag{4.3}$$

Finally, rule (14) computes $P(t_3|t_1, t_2)$ factor, because t_1 and t_2, are the unique fathers of t_3, and rule (8) computes $P(t_1)$ factor because $\mathit{root}(t_1)$ holds.
Concerning $P(t_2)$ factor in equation 4.1, it also can be computed applying rule (8) because $\mathit{root}(t_1)$ holds.
As equation 4.1 shows, we have also to compute the normalisation factor $\mathcal{K}$. In order to do this, it is necessary to compute $P(t_3|\bar{t}_2)$, $P(\bar{t}_2)$, and $P(t_3|t_1, \bar{t}_2)$. They are identical to the previous factors except for the truth value of t_2, and they are computed in the same way.

Finally, reordering all the previous directly computable factors, all the applied rules permit us to solve the bayesian query in the following way:

$$P(t_2|t_6) = \frac{P(t_6|t_3)\cdot P(t_3|t_1,t_2)\cdot P(t_1)\cdot P(t_2)}{(P(t_6|t_3)\cdot P(t_3|t_1,t_2)\cdot P(t_1)\cdot P(t_2))+(P(t_6|t_3)\cdot P(t_3|t_1,\bar{t}_2)\cdot P(t_1)\cdot P(\bar{t}_2))}.$$

4.4 Inference over polytrees

In this section we describe the rules concerning inference over polytrees. From the ontology point of view, it means we can formulate queries involving object properties. At this level each arc of the polytree has a specific semantic given by the object property it refers to, and a label given by the name of that object property. So, the conditional probability of an HLN, depends on both the evidence node and what arcs bind it to the evidence node.
As discussed in chapter 3, each arc has its probability distribution related to the domain and the range of the object property that it refers, but it also induces its conditioning to the other HLNs along a query path (we called it as induced conditional probability). In fact, crossing an arc p_i along a path means that we have to restrict the next conditioning sub-space, to that one in which the object property referred by p_i holds. In terms of space complexity, it can't be efficent to compute in advance all the conditional probabilities of all the possible conditionings among sub-spaces individuated by each arc. It is more convenient to compute dinamically all the induced conditional probabilities, every time we have to solving a specific bayesian query. Also for computing these probabilities, the Bayes formula is used, but the induced sub-space in which its arguments are evaluated, is depending on the set of arcs via which we reached the actual arc. In our operational semantic, first of all, each bayesian query is decomposed into the product of factors of directly computable probabilities, and then the rules computing each conditional probability factor are applied, taking into account the induced conditional probability.
Another aspect to evaluate is that a node can reach another node via different sets of arcs, here called $paths$, as stated in definition 24. It means that in the initial ontology there are many object properties which have

both the same domain and the same range. So, the path uniqueness property of the polytrees class, appears to be violated. In reality, it is sufficient that the path uniqueness property holds among the nodes involved in bayesian queries. Since, each query requires to specify the set of object properties composing the path binding the query node to the evidence node, the path uniqueness property trivially holds.
In order to specify the rest of the operational semantic of the BQ language, we need some additional notation.
In the following, each t_i node is a High Level Node (HLN) meaning that it represents a whole taxonomy i, as specified in Chapter 3.

Definition 27 *Let p_1, p_2 to be paths in $\mathcal{R}$ binding t to t_1 and t to t_2 respectively, with $t_1 \gg t$ and $t_2 \ll t$. $p = p_1.p_2$ is the set of paths binding t_1 to t_2.*

Notice that, since both p_1 and p_2 are unique paths, also p is a unique path.

Definition 28 *Let t_1 and p to be a node and a path, respectively, belonging to the polytree $\mathcal{R}$. The function*

$$t_1.p \Rightarrow t_2,$$

specifies that t_2 is the node reached by t_1 via p.

Definition 29 *Let $\{p_1, p_2, ..., p_n\}$ to be a set of paths belonging to the polytree $\mathcal{R}$, such that $p = p_1.p_2.p_3. \cdots .p_{n-1}.p_n$ is a path and $p \in \mathcal{R}$. We define the following functions:*

$$head(p) = p_1.p_2. \cdots .p_{n-1}$$

and

$$tail(p) = p_2.p_3. \cdots .p_n$$

Definition 30 *Let t be a node belonging to the polytree $\mathcal{R}$, and $\mathcal{DA}(t) = \{t_1^{\mathcal{DA}}, t_2^{\mathcal{DA}}, ..., t_m^{\mathcal{DA}}\}$ a set of m direct ancestors of t. The function*

$$arc(t, \mathcal{DA}(t), \mathcal{R}),$$

returns the set of the names of the object properties that each arc, binding t to each $t_i^{\mathcal{DA}}$ with $i = 1, ..., n$, refers.

Evidence both *under* and *over* w.r.t. query (1)

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(t_2|_{p_2} t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|_{p_1} t_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(t_2|_{p_2} \overline{t}, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\overline{t}|_{p_1} t_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t|_p(t_1, t_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

Evidence *over* w.r.t. query (2)

$$\frac{\begin{array}{c} t_2 \gg t_1, \exists p : (t_2.p \Rightarrow^* t_1) \\ \{t_0\} = \mathit{father}(t_1, \mathcal{R}), \{p_0\} = \mathit{arc}(t_1, \{t_0\}, \mathcal{R}), \exists j : t_j.\mathit{tail}(p) \Rightarrow^* t_2 \\ \mathcal{P}(t_1|_{\{p_0\}} \{t_0\}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_j|_{\mathit{tail}(p)} t_2, \mathcal{R}) \rightarrow n_2 \\ \forall i : ((t_i \in \{t_0\}) \wedge \overline{\mathit{ances}}(t_2, t_i, \mathcal{R}) \wedge (p_i = \mathit{arc}(t_1, t_i, \mathcal{R})).\mathcal{P}(t_i|_{p_i} \varepsilon, \mathcal{R}) \end{array}}{\mathcal{P}(t_1|_p t_2, \mathcal{R}) \rightarrow n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t_0\})-1)} n_k}$$

Evidence *under* w.r.t. query (3)

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}) \\ \exists k : (1 \leq k \leq \mathcal{C}(\{t_0\}))t_1.\mathit{head}(p) \Rightarrow^* t_k, p_l = \mathit{arc}(t_2, t_k, \mathcal{R}) \\ \mathcal{P}(t_2|_{p_l} t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|_{\mathit{head}(p)} t_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|_{\mathit{head}(p)} \overline{t}_1, \mathcal{R}) \rightarrow n_4 \\ \exists p_m : p = p_m.\mathit{tail}(p), \mathcal{P}(t_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\overline{t}_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(t_1|_p t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{(n_1 \cdot n_2 \cdot n_3) + (n_1 \cdot n_4 \cdot n_5)}}$$

Evidence *under* w.r.t. query (query is a father of evidence) (3.1)

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}), t_1 \in \{t_0\} \\ p_l = \mathit{arc}(t_2, t_1, \mathcal{R}), \mathcal{P}(t_2|_{p_l} t_1, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(t_2|_{p_l} \overline{t}_1, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\overline{t}_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t_1|_p t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{(n_1 \cdot n_2) + (n_3 \cdot n_4)}}$$

Figure 36: Semantic rules about *under* and *over* evidence

Rule (1) in figure 36 concerns the most general kind of query over poly-trees. Figure 37 shows an example of query involving evidence both under and over w.r.t. query node. Concerning paths involved in each bayesian query, we ignore the direction of each arc composing them, but we consider only the direction of the whole path. Conventionally, we assume that the direction of each path goes from the node which is on the left of the symbol "|", to the node which is on the right w.r.t. the same symbol.
In figure 37, we observe that the evidence over the query, that is t_1, is connected to query t_q, via $p_{over} = p_6.p_3$ path. Analogously, the evidence under the query, that is t_2, is connected to query t_q, via $p_{under} = p_{10}.p_{14}.p_{15}$ path. We call the path binding t_1 to t_2 as $p = p_1.p_2$.

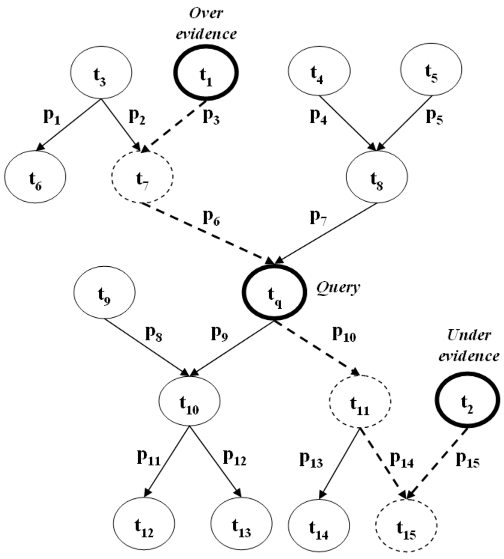


Figure 37: Example of evidence both under and over w.r.t. query.

We can separate those kinds of evidence in the computation of the bayesian query, by using the Bayes formula:

$$\mathcal{P}(t_q|_{p_{over},p_{under}} t_1, t_2) = \frac{\mathcal{P}(t_2|_{p_{under},p} t_q, t_1) \cdot \mathcal{P}(t_q|_{p_{over}} t_1)}{\mathcal{P}(t_2|_p t_1)}$$

As discussed in section 4.2, $\frac{1}{\mathcal{P}(t_2|_p t_1)}$ is a normalisation factor $\mathcal{K}$, and since t_q d-separates t_2 from t_1, we can write:

$$\mathcal{P}(t_q|_{p_{over},p_{under}} t_1, t_2) = \mathcal{P}(t_2|_{p_{under}} t_q) \cdot \mathcal{P}(t_q|_{p_{over}} t_1) \cdot \mathcal{K}$$

Semantic rules about both top-down and bottom-up inference over polytrees are very similar to those ones over taxonomy with multiple inheritance. In particular, all the steps involving d-separation property, Bayes formula, and polytrees properties, for constructing the semantic rules of BQ language, are the same as in section 4.3. So, in the following we show how the paths are involved in all the semantic rules, rather than how those rules have been constructed. We remark that the only difference is about each probability computation and not about the mathematical construction process of each rule. Here, we have to consider the semantics of each arc that composes each path, i.e., which are the arcs we are dealing with, at each step of the recursive processes. The reason for this, is that each arc refers to different ontology object properties, each of which has a specific semantics.

Rule (2) computes recursively the probability of each ancestor of the query node, given the evidence, till either an ancestor reaches the evidence node, or the evidence becomes $under$ the query of that ancestor. As discussed in section 4.3, at each step a bayesian query $\mathcal{P}(t_q|_p t_2)$ is decomposed into three factors:

(a) The probability of the query node t_q given all its fathers.

(b) The probability of the father of t_q which is connected to the evidence via p, given the evidence.

(c) The product of the a priori probabilities of the remaining fathers of t_q.

In figure 38 for example, the path binding t_q to t_2 is composed of three specific arcs, that are, p_6, p_2, and p_1. The first step is to involve the fathers of t_q. As rule (2) shows, we use the functions $father$ and arc. Related to figure 38, we obtain:

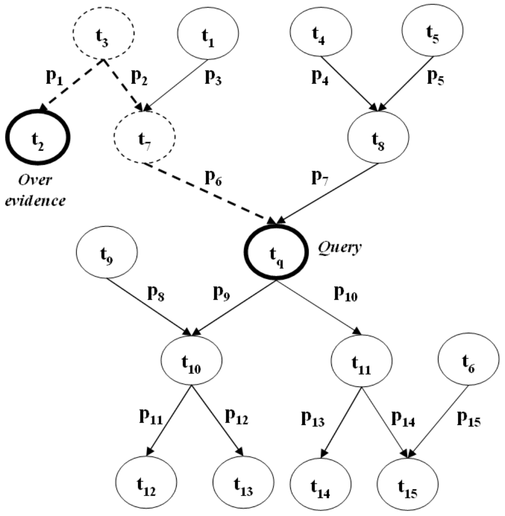


Figure 38: Example of evidence over w.r.t. query.

$$\textit{father}(t_q) = \{t_7, t_8\} \wedge \textit{arc}(t_q, t_7, t_8) = \{p_6, p_7\}$$

p_6 is the arc binding t_q to t_7, and p_7 is the arc binding t_q to t_8. So, concerning (a) we have:

$$\mathcal{P}(t_q|_{p_6,p_7}(t_7, t_8))$$

This probability is computed by rule (4), and we will esamine that rule in at the end of this section. Among all fathers of t_q, we have to select that father t_j by which we can reach the evidence. Since we are dealing with polytrees, the other fathers are independent from the evidence. In this respect, the rule verifies the following condition:

$$\exists j : t_j.\textit{tail}(p) \Rightarrow t_2$$

As you can see in figure 38, $j = 7$, and so concerning to (b), we have:

$$\mathcal{P}(t_7|_{\textit{tail}(p)}t_2)),$$

where $\textit{tail}(p) = p_2.p_1$ is the path binding t_7, the selected father of t_q, to the evidence.

Now we have to compute all the a priori probabilities of the remaining fathers of t_q, that are the fathers t_i for which the condition $\mathit{ancestor}(t_2, t_i)$ does not hold. We also need to know which are the arcs that bind each t_i to t_q. Related to our example, rule (2) checks the following condition:

$$\forall i : ((t_i \in \{t_7, t_8\}) \wedge \overline{\mathit{ances}}(t_2, t_i, \mathcal{R}) \wedge (p_i = \mathit{arc}(t_q, t_i, \mathcal{R})).\mathcal{P}(t_i|_{p_i}\varepsilon, \mathcal{R})$$

Since the only t_i that satisfies the condition is t_8, which is connected to t_q via p_7, concerning to (c), we have to compute the following probability:

$$\mathcal{P}(t_8|_{p_7}\varepsilon)$$

It is computed by rule (7) in figure 40. Note that this particular notation, means that, from the ontology point of view, we have to consider all the instances belonging to $\Phi^{-1}(t_8)$ which are related to the instances belonging to $\Phi^{-1}(t_q)$ by $\Psi^{-1}(p_7)$. In other terms, all the instances of $\Phi^{-1}(t_8)$ involved in the $\Psi^{-1}(p_7)$ relation.

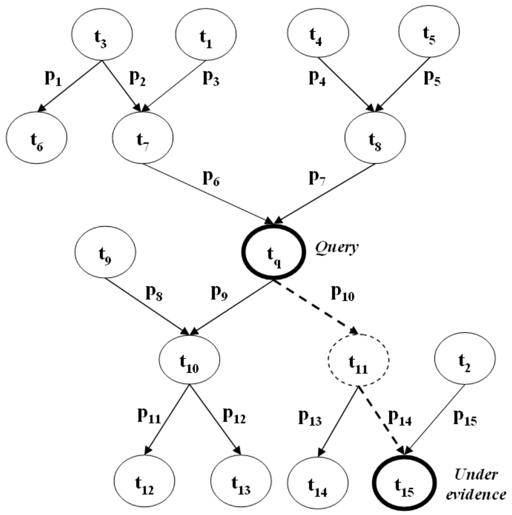


Figure 39: Example of evidence under w.r.t. query.

Rule (3) concerns the computation of each bayesian query when the evidence is *under* the query. In figure 39 is reported a graphical example of

this kind of query.
As discussed in section 4.3 in relation to rule $(3bis)$, a bayesian query is decomposed in the following three factors:

(a) The probability of the evidence node (t_{15} in figure 39) given the father of the evidence node which is descendent of the query node (t_{11} in figure 39).

(b) The probability of the previous father, given the query node.

(c) The a priori probability of the query node.

The main step is to apply Bayes formula to the bayesian query. Related to the example in figure 39, we obtain:

$$\mathcal{P}(t_q|_{p_{10}.p_{14}}t_{15}) = \mathcal{P}(t_{15}|_{p_{14}.p_{10}}t_q) \cdot \mathcal{P}(t_q|_{p_{10}}\varepsilon) \cdot \mathcal{K},$$

where $\mathcal{K} = \frac{1}{\mathcal{P}(t_{15}|_{p_{14}}\varepsilon)}$ is the normalisation factor. Note that the Bayes formula changes the query node with the evidence node, and consequently the path binding those nodes is implicitly reversed.
Now, we have to involve the descendent of the query node which is the father of the evidence node, that is t_{11}. In general, due to the d-separation property, the query node and the evidence node are independent each other, given the node satisfying the properties of t_{11}. So, in our case we obtain:

$$\mathcal{P}(t_{15}|_{p_{14}.p_{10}}t_q) = \sum_{t_{11}} \mathcal{P}(t_{15}|_{p_{14}}t_{11}) \cdot \mathcal{P}(t_{11}|_{p_{10}}t_q)$$

We can simply verify how rule (3) identifies those two factors, checking the conditions specified at the begin of the rule, in relation to figure 39:

$$\begin{gathered}
\mathit{father}(t_{15}, \mathcal{R}) = \{t_2, t_{11}\} \\
p = p_{10}.p_{14} \wedge \mathit{head}(p) = p_{10} \wedge \mathit{tail}(p) = p_{14} \\
\exists k.(t_q.p_{10} \Rightarrow t_k) \equiv t_{11} \\
p_l = \mathit{arc}(t_{15}, t_{11}, \mathcal{R}) \equiv p_{14} \\
\exists m.(p = p_m.\mathit{tail}(p)) \equiv p_{10}
\end{gathered}$$

$\mathcal{P}(t_{15}|_{p_{14}}t_{11})$ and $\mathcal{P}(t_{11}|_{p_{10}}t_q)$ correspond to (a) and (b), respectively. (c) is equal to $\mathcal{P}(t_q|_{p_{10}}\varepsilon)$ and it is computed by rule (7) in figure 40.

Probability distribution of a node given its $\mathcal{C}(\{t\})$ fathers (4)

$$\frac{\begin{array}{c}\{t\} = \mathit{father}(t,\mathcal{R}), \mathcal{S} = \mathcal{IC}(t,\mathcal{R}) \\ \forall i : i = 1, ..., \mathcal{C}(\{t\}).((t_i \in \{t\}) \wedge (p_i \in \{p\}) \wedge (p_i = \mathit{arc}(t, t_i, \mathcal{R}))) \\ n_2 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), \mathit{upper}(\Phi^{-1}(t))) \\ n_1 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), (\Phi^{-1}(t)) \models \mathcal{S})\end{array}}{\mathcal{P}(t|_p\{t\},\mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

Probability distribution reversed (5)

$$\frac{\begin{array}{c}\{t\} = \mathit{father}(t,\mathcal{R}) \\ \forall i : i = 1, ..., \mathcal{C}(\{t\}).((t_i \in \{t\}) \wedge (p_i \in \{p\}) \wedge (p_i = \mathit{arc}(t, t_i, \mathcal{R}))) \\ \exists i.(t_i \in \{t\}) \wedge (\mathcal{IC}(t_i) = \mathcal{S}) \\ n_1 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), (\Phi^{-1}(t))) \\ n_2 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), \mathit{upper}(\Phi^{-1}(t))) \\ \text{where } \exists k.(t_k \in \{t\}) \wedge ((\Phi^{-1}(t_k)) \models \mathcal{S})\end{array}}{\mathcal{P}(t|_p\{t\},\mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

A priori induced probability distribution (6)

$$\frac{\begin{array}{c}\mathcal{IC}(t) = \mathcal{S} \\ n_1 = \#_{inst}(\Phi^{-1}(t) \models \mathcal{S}) \\ n_2 = \#_{inst}(\mathit{upper}(\Phi^{-1}(t)))\end{array}}{\mathcal{P}(t|\varepsilon,\mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

A priori probability distribution (7)

$$\frac{\begin{array}{c}\mathit{root}(t,\mathcal{R}), \exists t_1.(t.p \Rightarrow t_1) \\ n_1 = \#_{triple}(\mathit{upper}(\Phi^{-1}(t)), \Psi^{-1}(p), \mathit{upper}(\Phi^{-1}(t_1))) \\ n_2 = \#_{inst}(\mathit{upper}(\Phi^{-1}(t)))\end{array}}{\mathcal{P}(t|_p\varepsilon,\mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

Figure 40: Semantic rules about induced conditioning

The process of computation of $\mathcal{K}$ is not very different from the one discussed in section 4.3. Similarly to $\mathcal{P}(t_q|_{p_{10}.p_{14}}t_{15})$, we have to compute $\mathcal{P}(\bar{t}_q|_{p_{10}.p_{14}}t_{15})$, for obtaining $\mathcal{P}(t_{11}|_{p_{10}}\bar{t}_q)$ and $\mathcal{P}(\bar{t}_q|_{p_{10}}\varepsilon)$. The computation process is identic to the previous one, but we have to consider $\bar{t}_q$ instead of t_q
Since $\mathcal{K} = \mathcal{P}(t_q|_{p_{10}.p_{14}}t_{15})\cdot\mathcal{P}(t_q|_{p_{10}}\varepsilon)+\mathcal{P}(\bar{t}_q|_{p_{10}.p_{14}}t_{15})\cdot\mathcal{P}(\bar{t}_q|_{p_{10}}\varepsilon)$, replacing it in the initial query, we obtain the following final computation:

$$\frac{\mathcal{P}(t_{15}|_{p_{14}}t_{11})\cdot\mathcal{P}(t_{11}|_{p_{10}}t_q)\cdot\mathcal{P}(t_q|_{p_{10}}\varepsilon)}{(\mathcal{P}(t_{15}|_{p_{14}}t_{11})\cdot\mathcal{P}(t_{11}|_{p_{10}}t_q)\cdot\mathcal{P}(t_q|_{p_{10}}\varepsilon))+(\mathcal{P}(t_{15}|_{p_{14}}t_{11})\cdot\mathcal{P}(t_{11}|_{p_{10}}\bar{t}_q)\cdot\mathcal{P}(\bar{t}_q|_{p_{10}}\varepsilon))},$$

as rule (3) shows.
Rule (3.1) in figure 36 is very similar to $(3bis)$ in figure 32, but it consid-

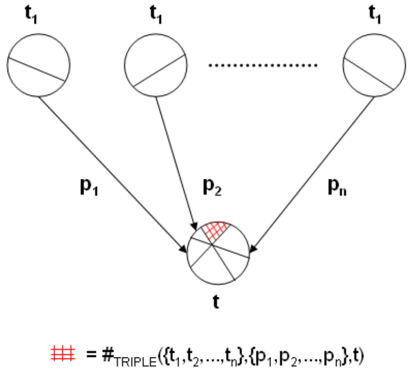


Figure 41: Example of *triple* function.

ers the case in which evidence is a direct descendent of query. Here, it is required only to apply the Bayes formula, in order to change the evidence from *under* to *over* w.r.t. query. Then, on that factor we have to apply rule (2) in figure 36. The computation of the other resulting factors, that are $\mathcal{K}$ and the a priori probability, is identical to the previous one.

Now, we describe the rules for computing the induced conditional probability. First, we introduce the following functions.

Definition 31 *Let t, $\{t_i\}$ be nodes belonging to a polytree $\mathcal{R}$, such that $\{t_i\}$ are*

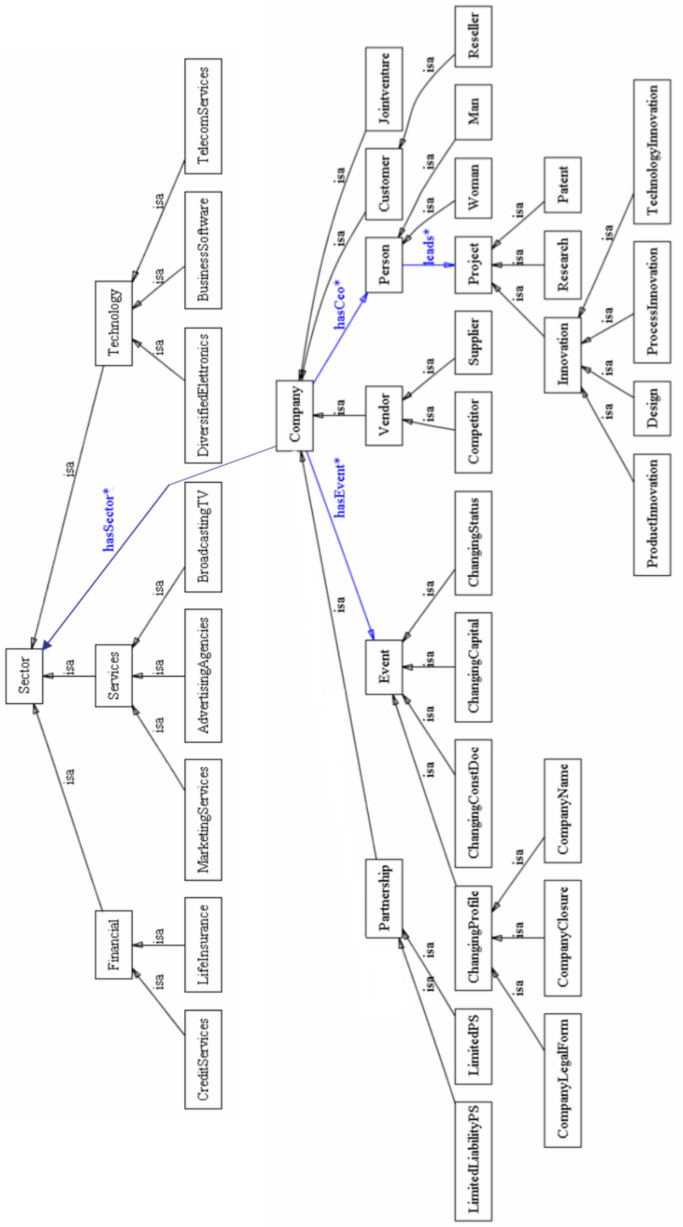


Figure 42: Example of Ontology.

direct ancestors of t. Let p_1, p_2, ..., p_n arcs belonging to $\mathcal{R}$, such that $\forall i.p_i = arc(t, \{t_i\})$ with $n = 1, ..., n$. Function $\#_{triple}$ is defined as:

$\#_{triple}$: (set of domains $\times$ set of properties $\times$ range class) $\longrightarrow$ number of instances

It considers all the properties among each domain class and the range class, and it counts all the instances of the range class, which have a relation with instances of those domains via all the properties, at the same time.

Figure 41 shows an example about $\#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), \Phi^{-1}(t))$. It counts all the instances of $\Phi^{-1}(t)$, individuated by the intersection of all $\Psi^{-1}(\{p_i\})$ properties.

Definition 32 *Let $\mathcal{IC}(t)$ be the function returning the sub-space of conditioning of t w.r.t. the initial query.*

$$\mathcal{IC}(t, \mathcal{R}) = \begin{cases} \oslash & \text{if } t \text{ is the initial query;} \\ \mathcal{S} & \text{otherwise.} \end{cases}$$

Definition 33 *Let t_i be a node representing a taxonomy class. Function upper(t_i) returns the HLN $\in \mathcal{R}$, to which t_i belongs.*

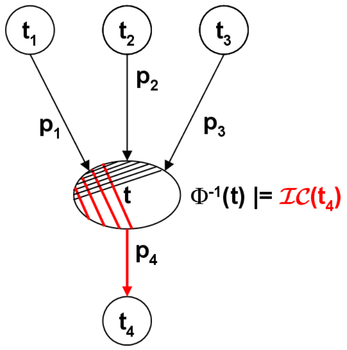


Figure 43: Example of a node with three father nodes

Rule (4) in figure 40 permits us to compute the probability of a node given all its $\mathcal{C}(\{t\})$ fathers. Figure 43 shows a graphical example, in which

t_1, t_2, t_3, are fathers of t and they are connected to it, via p_1, p_2, p_3, respectively. It means that in the initial ontology there are three object properties, that are $\Psi^{-1}(p_1)$, $\Psi^{-1}(p_2)$, $\Psi^{-1}(p_3)$. Each one binds instances of $\Phi^{-1}(t_1)$, $\Phi^{-1}(t_2)$, $\Phi^{-1}(t_3)$ classes respectively, to $\Phi^{-1}(t)$ class. Since function $\mathcal{IC}(t)$ returns a sub-space of conditioning $\mathcal{S}$, in this rule we have to consider all instances of t satisfying the sub-space individuated by $\mathcal{S}$:

$$(\Phi^{-1}(t) \models \mathcal{S}).$$

Every time we cross an arc in $\mathcal{R}$, we restrict our space in which we have to evaluate the terms of the next probability. In figure 43, for example, supposing we just crossed p_4 arc, we have to evaluate all the instances of $\Phi^{-1}(t)$, starting from the space of t_4, that are in relation with $\Psi^{-1}(p_4)$, $\Psi^{-1}(p_4)$, and $\Psi^{-1}(p_4)$ in the sub-space of t at the same time, and satisfying the property $\Psi^{-1}(p_4)$.
Now, first, we have to count all the instances of $\Phi^{-1}(t)$ belonging to the space individuated by $\mathcal{S}$, each of which is involved in $\Psi^{-1}(p_1)$, $\Psi^{-1}(p_2)$, $\Psi^{-1}(p_3)$ relations, at the same time:

$$n_2 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), (\Phi^{-1}(t)) \models \mathcal{S})$$

Then we have to consider the whole range of the properties, that is the HLN to which t could belong. So, we have to count all the instances of $upper(\Phi^{-1}(t))$ for which all the properties hold:

$$n_1 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), upper(\Phi^{-1}(t)))$$

The ratio between the second result and the first one gives the probability of t given all its fathers w.r.t. the specific relations among them.
The case in which there are no induced conditioning, i.e., t coincides with the initial query, is very similar to the previous one. The function $\mathcal{IC}(t)$ returns $\oslash$, so we can a priori evaluate all the instances of t. The related rule is reported in the appendix A.

4.4.1 Example of Inference over Polytrees

Here we show an example of Bayesian query. It refers to the ontology depicted in figure 42. It involves the concepts of *Project*, representing various kinds of projects, *Event* representing changes of status or profile of each company, *Sector*, representing the area in which each company operates, *Company*, and *Person*. All these classes are bound each other by the *hasCeo* object property, connecting *Person* with *Company*, *hasSector* connecting *Company* with *Sector*, *hasEvent* connecting *Company* with *Event*, and *leads* connecting *Person* with *Project*. The compiling process, described in the previous chapter, builds the two-level Bayesian network shown in figure 44. As you can see, each HLN, called T_i, is a bayesian network coding all $is - a$ ontology relations, and the arcs among HLN identify an upper level of the bayesian network, coding all ontology object properties. Over this structure we can solve all bayesian queries that can be written using the language BQ. Note that the nodes which our semantic rules refer to, sometimes can be implicitly HLNs. So, in the following, each t can refer either to a $node$ or to an HLNs depending on whether some conditions about a specific taxonomy are expressed or not. For example, if we are looking for instances of $Jointventures$, it means we have an implicit condition about the company taxonomy like Company=$Jointventure$. If we are interested in all kinds of companies we have no conditions over that taxonomy, and we can directly refer to the HLN related to company taxonomy. In general, t refers to a specific class of a specific taxonomy, T refers to the taxonomy to which that t class belongs.

Suppose we want to know the likelihood that research projects are part of a company's plans given that the involved companies operates in the technology sector [3]. In other terms, we want to know the probability that a path exists between *Research Project* and *Technology Sector*[4]. By using the language BQ, we specify those object properties that we want to involve in our bayesian query:

$$\mathcal{P}(ResearchProject|_{(leads.hasCeo.hasSector)}TechnologySector) \tag{4.4}$$

[3]Note that, there are no explicity conditions about $Person$, $Event$, and $Company$

[4]Note that, the path uniqueness property holds at the moment we specify each query. Even if many paths binding each other two HLNs exist, one specific path is selected at the inference time.

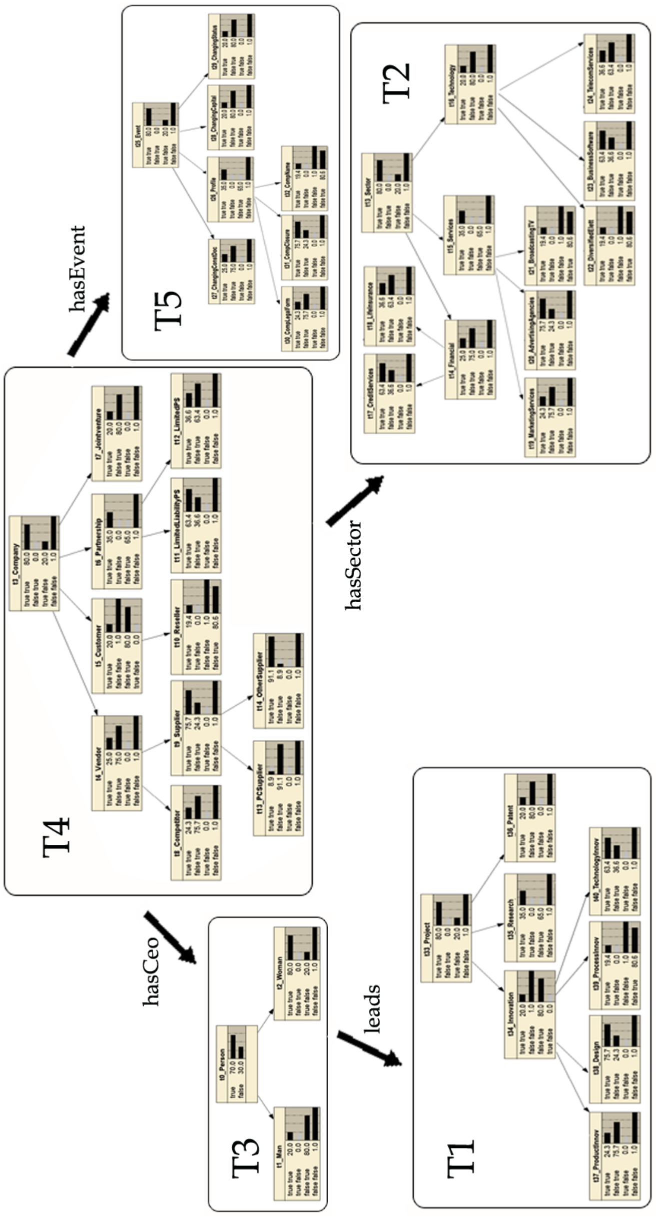


Figure 44: The ontology compiled into the $2lBN$.

Object property	**Bayesian arc**
hasCeo	p_1
hasSector	p_2
hasEvent	p_3
leads	p_4

Figure 45: Function Ψ maps each object property to a bayesian arc.

In terms of bayesian nodes the query running in our system is:

$$\mathcal{P}(t_{35}|_{(p_4.p_2.p_3)}t_{16}) \tag{4.5}$$

where $p = (p_4.p_2.p_3)$. All the correspondences among classes and nodes defined by Φ, and among object properties and arcs defined by Ψ, are shown in figure 46 and 45 respectively.
Now we examine the query. We note that only one evidence node is specified. So rules (2), (3), (3.1), (5), are possible. Since $T_2 \gg T_1$, where T_1 represents the HLN to which t_{35} belongs, and T_2 represents the HLN to which t_{16} belongs (as figure 44 shows), the remaining candidates rules, are rule (2) and (5). But T_2 is not a father of T_1, so we have to apply rule (2). Now, checks are required for arcs and nodes satisfying the conditions in the head of the rule:

- for $\{T_0\} = \mathit{father}(T_1, \mathcal{R})$ we obtain T_3,

- for $\{p_0\} = \mathit{arc}(T_1, T_3, \mathcal{R})$ we obtain p_4, and

- for $\exists j : T_j.\mathit{tail}(p) \Rightarrow^* T_2$ we obtain $j = 3$ and T_3, since $\mathit{tail}(p) = p_2.p_3$

Since T_3 is the only father of T_1, i.e., $\mathcal{C}(\{T_0\}) = 1$, the condition $\overline{\mathit{ances}}$ in rule (2) can not be applied on any HLN, and we consider the factor $\prod_{k=1}^{0} n_k$ equal to one. On the basis of the previous conditions, the rule decomposes the intial query into the product of two other probabilities[5]:

$$\mathcal{P}(t_{35}|_{(p_4)}T_3) \cdot \mathcal{P}(T_3|_{(p_2.p_3)}t_{16}) \tag{4.6}$$

[5]For semplicity of notation, in the examples we omit each probability factor involving the same nodes with a different thrut values.

Note that t_{35} is the original query, that is a specific class of the *Project* taxonomy, and t_{16} is the original evidence, that is a specific class of the *Sector* taxonomy. When we consider all the other nodes along the query path, we refer to them as HLNs, since there are no conditions on those taxonomies. Just to give a more intuitive idea of what we obtained by using the rule, we re-write the equation 4.6 with reference to the ontology entity names:

$$\mathcal{P}(ResearchProject|_{(leads)}Person) \cdot \mathcal{P}(Person|_{(hasCeo.hasSector)}TechnologySector) \tag{4.7}$$

Rule (4) is applied to the first factor of equation 4.6, because HLN T_3 is

Ontology class	Bayesian node	Ontology class	Bayesian node
Person	t_0	*BroadcastingTV*	t_{21}
Man	t_1	*DiversifiedElett*	t_{22}
Woman	t_2	*BusinessSoftware*	t_{23}
Company	t_3	*TelecomServices*	t_{24}
Vendor	t_4	*Event*	t_{25}
Customer	t_5	*ChangingProfile*	t_{26}
Partnership	t_6	*ChangingConstDoc*	t_{27}
Jointventure	t_7	*ChangingCapital*	t_{28}
Competitor	t_8	*ChangingStatus*	t_{29}
Supplier	t_9	*CompanyLegalForm*	t_{30}
Reseller	t_{10}	*CompanyClosure*	t_{31}
LimitedLiabilityPS	t_{11}	*CompanyName*	t_{32}
LimitedPS	t_{12}	*Project*	t_{33}
Sector	t_{13}	*Innovation*	t_{34}
Financial	t_{14}	*Research*	t_{35}
Services	t_{15}	*Patent*	t_{36}
Technology	t_{16}	*ProductInnov*	t_{37}
CreditServices	t_{17}	*Design*	t_{38}
LifeInsurance	t_{18}	*ProcessInnov*	t_{39}
MarketingServices	t_{19}	*TechnologyInnov*	t_{40}
AdvertisingAgencies	t_{20}	-	-

Figure 46: Function Φ maps each ontology class to a bayesian node.

the unique father of T_1, via path p_4 [6]. Such rule verifies if an induced conditioning over t_{35} exists. Function $\mathcal{IC}(t_{35})$ returns the empty set, because t_{35} is the initial node of the path, and we do not have to cross any arc for reaching it. So, that probability is computed starting from all the sub-space of the research projects.

Rule (2) is recursively applied to $\mathcal{P}(T_3|_{(p_2.p_3)}t_{16})$ factor with $p = p_2.p_3$, be-

[6]Note that T_1 represents the HLN to which t_{35} belongs.

cause $T_2 \gg T_3$. As shown before, it checks for arcs and nodes satisfying the conditions in the head of the rule:

- for $\{T_0\} = \mathit{father}(T_3, \mathcal{R})$ we obtain T_4,

- for $\{p_0\} = \mathit{arc}(T_3, T_4, \mathcal{R})$ we obtain p_1, and

- for $\exists j : T_j.\mathit{tail}(p) \Rightarrow^* T_2$ we obtain $j = 4$ and T_4, since $\mathit{tail}(p) = p_2.p_3$

So, the second factor of equation 4.6, is decomposed into the following factors:

$$\mathcal{P}(T3|_{(p_1)}T_4) \cdot \mathcal{P}(T4|_{(p_2)}t_{16}) \tag{4.8}$$

In terms of ontology entities names, we have:

$$\mathcal{P}(Person|_{(hasCeo)}Company) \cdot \mathcal{P}(Company|_{(hasSector)}TechnologySector) \tag{4.9}$$

On the first factor of equation 4.8, is applyed rule (4), because T_4 is the unique father of T_3, and they are connected each other via p_1 path. Rule (4) verifies if exists an induced conditioning. Function $\mathcal{IC}(T_3)$ returns a space $\mathcal{S} \neq \oslash$, because for reaching T_3 node we have to cross p_1 arc. It means that we restricted T_3 sub-space to all elements belonging to T_3 but also satisfying p_1 relation. In terms of the semantic given by the ontology, we restrict the sub-space of all person, that is T_3, to that of the person leading research projects.
Now, analyse the second factor of equation 4.8. Now, rule (2) is not anymore applicable because the evidence node T_2 is passed $under$ the query node T_4. Since $T_2 \ll T_4$, we have two candidates rules to be applied, that are rule (3) and rule (3.1). Since T_4 is the father of T_2, our system applies rule (3.1), and checks for the following conditions:

- for $\{T_0\} = \mathit{father}(T_2, \mathcal{R})$ we obtain T_4, and

- for $\{p_0\} = \mathit{arc}(T_2, T_4, \mathcal{R})$ we obtain p_2

So, the factor is decomposed into the following terms:

$$\frac{\mathcal{P}(t_{16}|_{(p_2)}T_4) \cdot \mathcal{P}(T_4)}{\mathcal{P}(t_{16}|_{(p_2)}T_4) \cdot \mathcal{P}(T_4) + \mathcal{P}(t_{16}|_{(p_2)}\overline{T}_4) \cdot \mathcal{P}(\overline{T}_4)} \tag{4.10}$$

On $\mathcal{P}(t_{16}|_{(p_2)}T_4)$, is applyed rule (5) because T_4 is the father of T_2 to which t_{16} belongs. We note that having applied Bayes formula, the induced sub-space is became the evidence, instead of the query. Rule (5) is aware of query and evidence have been reversed, and compute the probability in way very similar to rule (4). $\mathcal{P}(t_{16}|_{(p_2)}\overline{T}_4)$ is the same factor with different thrut values and it is computed in the same way, by the rule (2.2) reported in appendix A.
$\mathcal{P}(T_4)$ is a prior probability involving no arcs, and it is computed by rule (6). $\mathcal{P}(T_4)$ derives from the second factor of equation 4.8, in which T_4 has an induced conditioning. So, also in $\mathcal{P}(T_4)$, T_4 has to be evaluated awaring of its induced conditioning. It represents the sub-space of the companies having CEOs which lead research projects. $\mathcal{P}(\overline{T}_4)$ is computed in a similar way by rule (6.1) in appendix A.
Reordering all the computed factors, and replacing them in equation 4.5, we obtain the following final computation:

$$\frac{\mathcal{P}(t_{35}|_{(p_4)}T_3) \cdot \mathcal{P}(T_3|_{(p_1)}T_4) \cdot \mathcal{P}(t_{16}|_{(p_2)}T_4) \cdot \mathcal{P}(T_4)}{\mathcal{P}(t_{16}|_{(p_2)}T_4) \cdot \mathcal{P}(T_4) + \mathcal{P}(t_{16}|_{(p_2)}\overline{T}_4) \cdot \mathcal{P}(\overline{T}_4)} \tag{4.11}$$

In terms of the ontology semantic we have:

$$\frac{\mathcal{P}(ResProj|_{(leads)}Pers) \cdot \mathcal{P}(Pers|_{(hasCeo)}Comp) \cdot \mathcal{P}(TechSec|_{(hasSec)}Comp) \cdot \mathcal{P}(Comp)}{\mathcal{P}(TechSec|_{(hasSec)}Comp) \cdot \mathcal{P}(Comp) + \mathcal{P}(TechSec|_{(hasSec)}\overline{Comp}) \cdot \mathcal{P}(\overline{Comp})} \tag{4.12}$$

4.5 System Architecture

Figure 47 shows the architecture of the system in a simple schematic way. First of all, starting from the ontology, the *Ontology Compiling Module* constructs the two-levels Bayesian network as described in chapter 3, over which making inference. Then, for each query specified by the user by using the $GUI\ Module$, the $Query\ Checker\ Module$ verifies the syntactical correctness of the query, according to the BQ grammar defined in section 4.1. If the query belongs to the BQ language, it is processed by the $Inference\ Module$. This module recognizes the kind of query as shown in table 2. There, a fragment of the pseudo-code related to the query recognition is reported. By using the function $getEvidence(.)$, we are able to extract the evidence part from the Bayesian Query. If it returns $\emptyset$, represented by the symbol "E" (empty) at line 5, we have to deal with a prior probability. Otherwise the structure $evidence$ contains the evidence of the Bayesian query which can be single (evidence under or evidence over) or double (evidence under and over), as shown at lines 8-15. All

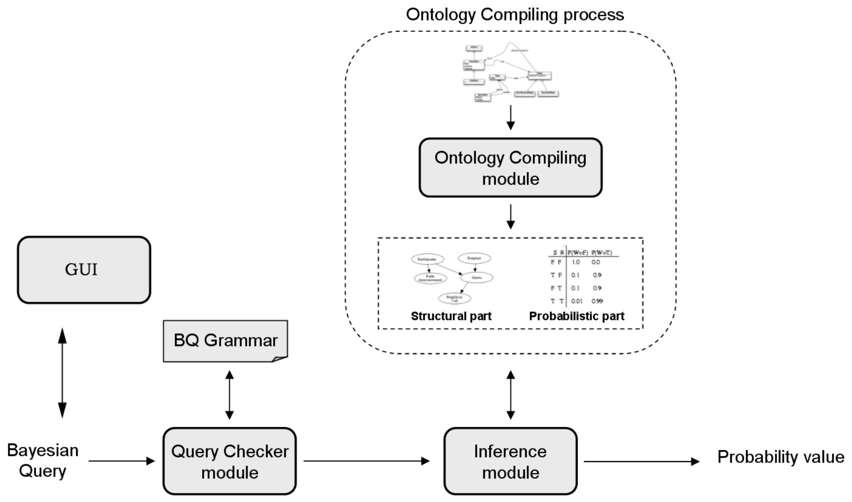


Figure 47: System Architecture.

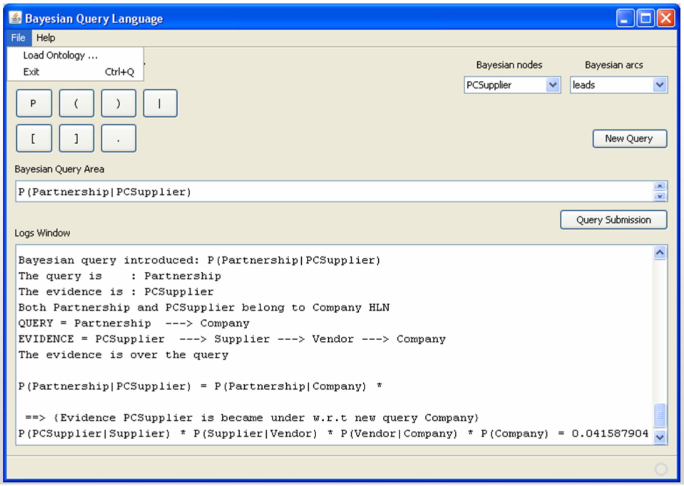
the modules are implemented in Java. In the following we show some details of each architecture module.

4.5.1 GUI Interface Module

Figure 48 shows the GUI interface of the System. It is composed of five parts. The first one allows the user to load an ontology for extracting

```
$real$ **BayesianQuery**$(\mathcal{BQ})$ {
    $query = \mathcal{BQ}.getQuery();$
    $evidence = \mathcal{BQ}.getEvidence();$
    $switch(evidence.getType())$ {
        $case < E >:$
            $result =$ **prior**$(query);$
            $break;$
        $case < U >:$
            $result =$ **under**$(query, evidence[0]);$
            $break;$
        $case < O >:$
            $result =$ **over**$(query, evidence[0]);$
            $break;$
        $case < UO >:$
            $result =$ **underover**$(query, evidence[0], evidence[1]);$
            $break;$
    }
    $return\ result;$
}
```

Table 2: General procedure for detecting the kind of query.



```
P(Partnership|PCSupplier)
```

```
Bayesian query introduced: P(Partnership|PCSupplier)
The query is    : Partnership
The evidence is : PCSupplier
Both Partnership and PCSupplier belong to Company HLN
QUERY = Partnership  ---> Company
EVIDENCE = PCSupplier  ---> Supplier ---> Vendor ---> Company
The evidence is over the query

P(Partnership|PCSupplier) = P(Partnership|Company) *

 ==> (Evidence PCSupplier is became under w.r.t new query Company)
P(PCSupplier|Supplier) * P(Supplier|Vendor) * P(Vendor|Company) * P(Company) = 0.041587904
```

Figure 48: System GUI.

the Bayesian network by applying the ontology compiling process. The second part allows the user to view all the Bayesian nodes and Bayesin arcs of the extracted $2lBN$, scrolling them in the proper combo boxes. The third part is an useful keypad for composing Bayesian queries; when an user specifies Bayesian queries referring to HLNs, the path connecting the query HLN with the evidence HLN must be enclosed in square brackets and it must follow the Bayesian conditioning symbol. Each sequence of one or many object properties, separated by a dot, identifies a path. The fourth part is the $Bayesian\ Query\ Area$ where an user can specify its Bayesian query. Finally, all the output messages, the query results, and the possible errors are reported in the last part of the GUI, that is the $Logs\ Window$.

4.5.2 Query Checker Module

This module checks the syntax of the Bayesian query. It consists in verifying if the query belongs to the language generated by the BQ grammar defined at the begin of this chapter. However, from the syntactical point of view, the user is guided in composing queries because both node and arc names are selectionable from the combo boxes in the part two of the GUI, and the syntactical symbols of the Bayesian query structure can be specified by using the GUI keypad.

4.5.3 Ontology Compiling Module

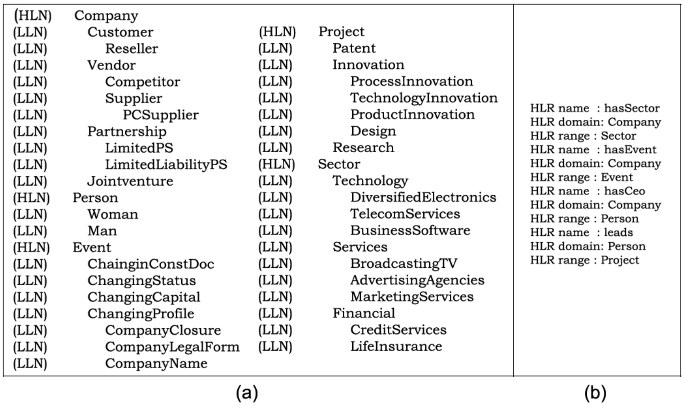
This module performs the $2lBN$ construction task. As described in chapter 3, this task involves both the construction of the Bayesian arcs and nodes, and the computation of the initial probability distributions. Regarding to the first one, we recursively visit the ontology $is - a$ structure in order to extract the LLNs and then we construct the HLNs. Then, we examine how the ontology classes are related each other by the object properties, in order to construct the Bayesian arcs connecting HLNs. Table 3 reports the pseudo code related to this part.

The $BayesianNodes$ function at line 20 loads the ontology. We start from the class $Thing$, that is the root class of each ontology. The vari-

```
void BayesianNodes(onto, class) {
   reasoner.loadOntologies(Collection.singleton(onto));
   detectBayesNodes(class, 0);
   reasoner.clearOntologies();
   }

void detectBayesianNodes(class, level) {
   // Recursive visit to the structure of the ontology //
   if (level == 1) {
    out.print(HLN);
    }
   else {
    if (level! = 0 {
     out.print(LLN);
    }
   }
   for (i = 0; i < level * INDENT; i + +) {
    out.print();
    }
   out.println(labelFor(clazz));
   // Find the children and recurse //
   Set < Set < OWLClass >> children = reasoner.getSubClasses(clazz);
   for (Set < OWLClass > setOfClasses : children) {
    for (OWLClasschild : setOfClasses) {
     if (!child.equals(clazz)) {
      save node and its father
      detectBayesianNodes(child, level + 1);
     }
    }
   }
  }
}
```

Table 3: Recursive visit to the structure of the ontology.

able $level$ represents the depth level of each class within the ontology; $Thing$ has level equal to zero. The $detectBayesianNodes$ function at line



(HLN)	Company		
(LLN)	Customer	(HLN)	Project
(LLN)	Reseller	(LLN)	Patent
(LLN)	Vendor	(LLN)	Innovation
(LLN)	Competitor	(LLN)	ProcessInnovation
(LLN)	Supplier	(LLN)	TechnologyInnovation
(LLN)	PCSupplier	(LLN)	ProductInnovation
(LLN)	Partnership	(LLN)	Design
(LLN)	LimitedPS	(LLN)	Research
(LLN)	LimitedLiabilityPS	(HLN)	Sector
(LLN)	Jointventure	(LLN)	Technology
(HLN)	Person	(LLN)	DiversifiedElectronics
(LLN)	Woman	(LLN)	TelecomServices
(LLN)	Man	(LLN)	BusinessSoftware
(HLN)	Event	(LLN)	Services
(LLN)	ChainginConstDoc	(LLN)	BroadcastingTV
(LLN)	ChangingStatus	(LLN)	AdvertisingAgencies
(LLN)	ChangingCapital	(LLN)	MarketingServices
(LLN)	ChangingProfile	(LLN)	Financial
(LLN)	CompanyClosure	(LLN)	CreditServices
(LLN)	CompanyLegalForm	(LLN)	LifeInsurance
(LLN)	CompanyName		

(a)

HLR name : hasSector
HLR domain: Company
HLR range : Sector
HLR name : hasEvent
HLR domain: Company
HLR range : Event
HLR name : hasCeo
HLR domain: Company
HLR range : Person
HLR name : leads
HLR domain: Person
HLR range : Project

(b)

Figure 49: (a) HLNs and LLNs. (b) Bayesian arcs (HLRs).

25, is a recursive one that visits the $is - a$ ontology structure, every time calling itself over the children of the class that it is visiting (line 46) and stores each class and its father in an appropriate data structure (line 45). At lines 26-38 we prepare the visualization of the $2lBN$ structure as figure 49(a) shows. This is related to the ontology presented in the previous section. The Bayesian arcs (figure 49(b)) are retrieved in a simpler way, asking for the object properties of each ontology class. The probabilistic part of the $2lBN$, is not computed when its structural part is extracted, but it is calculated at Bayesian query computation time. Actually, the whole probabilistic part is not even computed but, for each query, the necessary probability distributions are computed at run time. After the $Inference\ Module$ has decomposed the query into the product of n probabilities $P_1, ...P_n$ with $n \geq 1$, each of which belonging to the set of the initial distributions, this module computes the values of those P_i. In order to do this, ontologies are stored in main memory. Ontologies are managed by Sesame, a Java platform for working with the Resource Description Framework (RDF). An important component of the Sesame architecture is the Storage And Inference Layer (Sail). API is a low level

System API (SPI) for RDF stores and inferences. Its purpose is to abstract from the storage and inference details, allowing various types of storage and inference to be used. In order to retrieve the ontology instances individuated by the bayesian queries, we perform queries to the ontology by using SPARQL. It stands for the SPARQL Protocol and RDF Querying Language. The Sesame server is an implementation of the SPARQL Protocol and is being used when uploading or exporting data from the repository.
This is very efficient in terms of space complexity, because we have not to use very large data structures for storing all the initial probabilities values. Besides, every time the ontology is populated by new instances, we have not to recompute the set of the initial distributions.

4.5.4 Inference Module

The $Inference\ Module$ implements all the rules defined in section 4.1.3 and reported in appendix A. Also, it interacts with the $ABox$ of the ontology in order to compute function $\mathcal{IC}$ at query computation time. Table 4 shows the pseudo-code related to a small subset of those rules. It refers to queries not involving Bayesian arcs among HLNs. Pseudo-code related to inference involving HLNs is conceptually very similar, but it requires to introduce the paths referring the object properties. Paths are retrieved from the Bayesian query by the function $getPath(.)$, and they are managed by the functions $headPath(.)$, $tailPath(.)$, and $HLRname(.)$ as the rules defined in section 4.4 show. Function **prior**(q) at line 52, returns the probability that an ontology instance belongs to a specific Low Level Node q, within a specific High Level Node D_c.
Since this pseudo-code fragment deals with taxonomies as described in section 4.2, each LLN can have at most one father. So, if q is the root node in D_c, the probability is the ratio between the ontolgy instances that have type q and all the ontology instances (lines 53-55). Otherwise we select the father of q and we compute the percentage of the ontology instances that have type q in the sub-space of the instances having the same type of the father of q.

Function **underover** splits the Bayesian query into the product of a probability with an evidence *under* the query, and another one with an evidence *over* the query (line 64), according to the BQ operational semantics defined in the previous sections.
Functions **under** and **over** are recursive, and they can call each other. Here, only the recursive call aspect is shown, that is how these two functions can interact each other; all the other features of the implementation are left out. Regarding to the function **under**, if the query q corresponds to the father of the evidence e_-, the function terminates given that it is possible to compute that probability. Otherwise, as line 70 shows, the function **under** calls the function **over**. It is very important to note that, according to the BQ operational semantics, the arguments passed to the function **over** are inverted, transforming a top-down inference schema into a bottom-up inference one. Function **over** terminates when evidence e_+ is the father of the query q. Otherwise, until the evidence remains *over* the query, the function calls recursively itself, where the new query q' is the father of q which occurs in the path binding q to e_+ (lines 79-80). It can happen that the evidence e_+ is not *over* q' anymore. In that case the function **over** calls the function **under** (line 76).

4.5.5 First Experimental Results

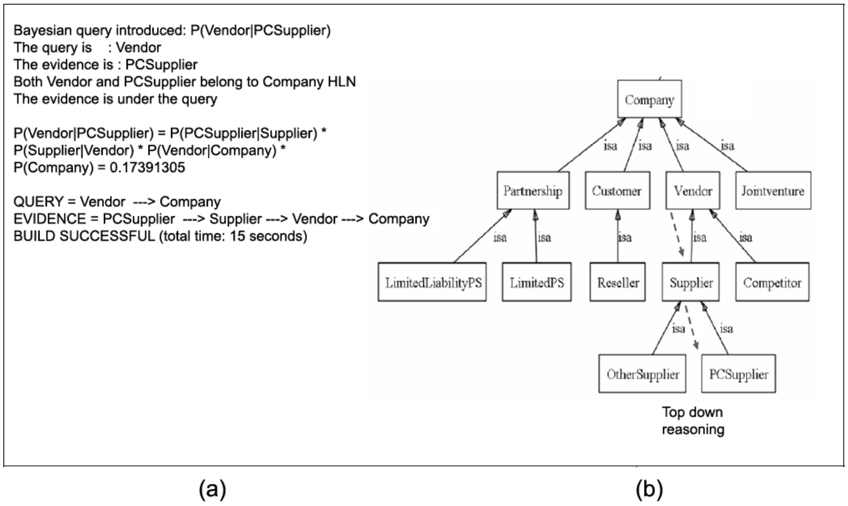
In this section we report some runs of our system by using the ontology presented in our thesis. The ontology compiling process produces a two-level Bayesian network with 41 Bayesian nodes (5 High Level Nodes and 36 Low Level Nodes), and 36 bayesian arcs (4 High Level Relations and 32 Low Level Relations). The rules involving the conjunction of evidences are not yet implemented. Furthermore, the results of the queries are presented except for their normalisation factors.
Concerning inference over taxonomies, we report the result about the following query:

Which is the probability of a Vendor instance to be a Supplier of PCs?

Figure 50(a) shows the result of the query and 50(b) shows what inference schema has been applied. From the logical point of view, this query com-

```
$real$ prior$(q)$ {
    $// prior\ probability\ related\ to\ is-a\ relation\ within\ a\ domain\ concept\ D_c\ (HLN)\ //$
    $if\ root(q, \mathcal{R}_{D_c})$ {
        $n = \frac{\#_{inst}(q, \mathcal{R}_{D_c})}{\#_{inst}(\mathcal{A}ll, \mathcal{R})}$;
    }
    $else$ { $//\ q\ has\ only\ one\ father\ //$
        $n = \frac{\#_{inst}(q, \mathcal{R}_{D_c})}{\#_{inst}(father(q, \mathcal{R}_{D_c})}$;
    }
    $return\ n$;
}

$real$ underover$(q, e_+, e_-)$ {
    $//\ \mathcal{K}\ is\ the\ normalisation\ factor\ //$
    $//\ q\ and\ e_-\ belong\ to\ LLN\ D_c\ //$
    $return\ (\mathcal{K} \cdot$ under$(q, e_-) \cdot$ over$(q, e_+))$;
}

$real$ under$(q, e_-)$ {
    $//\ \mathcal{K}\ is\ the\ normalisation\ factor\ //$
    $//\ q\ and\ e_-\ belong\ to\ LLN\ D_c\ //$
    $if\ (father(e_-, \mathcal{R}) == q)\ return\ (P(e_-|q) \cdot \mathcal{K} \cdot P(q))$;
    $else\ return\ (\mathcal{K} \cdot P(q) \cdot$ over$(e_-, q))$
}

$real$ over$(q, e_+)$ {
    $//\ q\ and\ e_+\ belong\ to\ LLN\ D_c\ //$
    $if\ (father(q, \mathcal{R}) == e_+)\ return\ P(q|e_+)$;
    $else$ {
      $if\ (e_+ \ll q)\ return$ under$(q, e_+)$;
      $else$ {
        $//\ q'\ is\ along\ the\ path\ connecting\ q\ to\ e_+\ //$
        $q' = father(q)$;
        $return\ (P(q|q') \cdot$ over$(q', e_+))$
      }
    }
}
```

Table 4: Procedure for *under*, *over*, and *under_over* evidence.



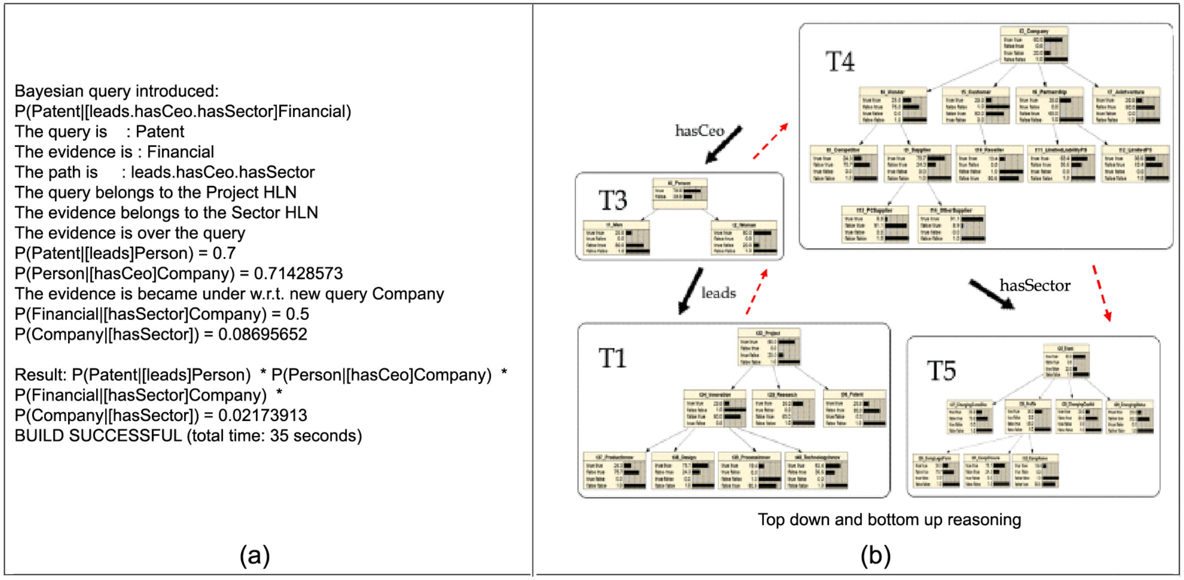
```
Bayesian query introduced: P(Vendor|PCSupplier)
The query is    : Vendor
The evidence is : PCSupplier
Both Vendor and PCSupplier belong to Company HLN
The evidence is under the query

P(Vendor|PCSupplier) = P(PCSupplier|Supplier) *
P(Supplier|Vendor) * P(Vendor|Company) *
P(Company) = 0.17391305

QUERY = Vendor  ---> Company
EVIDENCE = PCSupplier  ---> Supplier ---> Vendor ---> Company
BUILD SUCCESSFUL (total time: 15 seconds)
```

Figure 50: (a) Query result (b) Inference schema.

putes the inclusion degree of SupplierPC class in Vendor class.
With our system is possible to make infernce about object properties, and not only about classes inclusion degree, and classes overlap. So, we report the results about the following query, concerning inference involving object properties:

Which is the probability that a Patent project is led by person which is CEO of a company operating in the financial sector?

Figures 51(a) and 51(a) show the result of the query and the inference schema applied, respectively. As we can observe, it is necessary an $explaining$ $away$ inference schema, because during the inference process the evidence, which is initially $over$ the query, becomes $under$ the query, at the last recursive step.



```
Bayesian query introduced:
P(Patent|[leads.hasCeo.hasSector]Financial)
The query is    : Patent
The evidence is : Financial
The path is     : leads.hasCeo.hasSector
The query belongs to the Project HLN
The evidence belongs to the Sector HLN
The evidence is over the query
P(Patent|[leads]Person) = 0.7
P(Person|[hasCeo]Company) = 0.71428573
The evidence is became under w.r.t. new query Company
P(Financial|[hasSector]Company) = 0.5
P(Company|[hasSector]) = 0.08695652

Result: P(Patent|[leads]Person)  * P(Person|[hasCeo]Company)  *
P(Financial|[hasSector]Company)  *
P(Company|[hasSector]) = 0.02173913
BUILD SUCCESSFUL (total time: 35 seconds)
```

Figure 51: (a) Query result (b) Inference schema.

Chapter 5

Conclusion and Future Work

In general dealing with uncertainty is crucial in ontology tasks as domain modeling and ontology reasoning. As discussed in section 2.3, many proposals are focused on the representation of the uncertainty in ontology, and the reasoning over the enriched knowledge. Instead, our goal is to provide a query language for answering queries involving probabilities, by integrating ontologies with typical features of Bayesian networks, but without modifying the original ontology. The uncertainty is directly derived from the ontology explicit knowledge, and it is not necessary to know an initial probability distribution a priori. In this research we developed a Bayesian compiling process which has an ontology as input, in terms of its $TBox$ and $ABox$, and a two-level Bayesian network ($2lBN$) and its initial probability distributions as output. $2lBN$ permits to represent both the ontology taxonomical aspect and the relationships among ontology classes, that are object properties. Then, we defined a Bayesian query language which extends ontology queries with Bayesian network reasoning, in order to make inference over the two-level Bayesian network, involving both ontology $is-a$ relations and ontology object properties.

However there are a number of issues still to be addressed, which lead to several possible future works:

1. Handling ontology relationships
2. Dealing with logical relations between concepts introduced by OWL-DL
3. Bayesian query language extensions
4. Dealing with cycles in $2lBN$

We will very briefly discuss each of these issues next.

1. Handling ontology relationships

In our work we use a two-levels Bayesian network for representing a $TBox$ ontology. Each ontology class is mapped to a random Boolean variable, to find out all the domain concepts, and each of them is mapped to a random multi-value variable. So, at the upper level of the network each High Level Node (HLN) represents a domain concept of the ontology, and each labelled arc represents a specific object property. It is very important to note that each ontology object property between two classes c_i and c_j which are not root classes in their concept domains D_h and D_k, is represented as a labelled arc between the HLN of D_h and the HLN of D_k, in the same way if c_i and c_j are root classes in D_h and D_k, respectively. It means that the $2lBN$ structural part is the same in both cases, that is, each object property is represented as an arc binding only root classes. Then, in the $2lBN$ probabilistic part, for all the object properties between two specific classes that do not hold, their existence probability value will be equal to zero. Figure 52 shows an example. p connects D_h to D_k, but the object property that it refers to in the ontology binds c_i ontology class to c_j ontology class. So, for example, the probability that p binds instances of c_2 to instances of c_4 is equal to zero. An ontology in OWL can define object properties, but it permits to describe knowledge also by using data properties. They represent attributes of each concept of a domain. A very interesting future research, is the possibility to introduce data properties into the two-levels Bayesian network compiling process. It permits

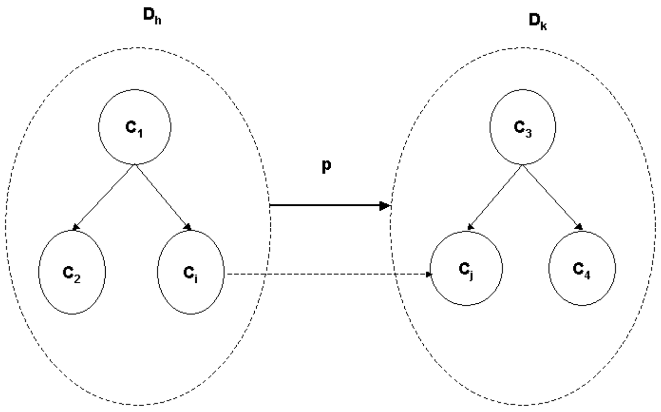
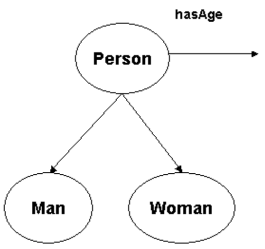


Figure 52: Object property binding c_i to c_j

to make inference over a richer Bayesian network, and to ask queries involving both object and data properties. For example, still referring to the examples in chapter 4, we could formulate a query like the following:

Which is the likelihood of default of a company having payments of capital in cash by amount superior to 300,000 Euros when such amount surpasses the 25% of the initial capital, given that it has a male older than 70 years, as chief of executive officer?

The problem is how to represent data properties in $2lBN$s, both in terms of $2lBN$ structural part and $2lBN$ probabilistic part. Since obviously we are interested in quantitative data properties, first of all we need to discretize all the ranges of all attributes that each data property refers to. From the structural point of view, we could represent each data property as a Low Level Node (LLN) $prop_{data}$, and connecting all the classes on which that data property is defined, to $prop_{data}$. Consequently, the probability tables for each data property could look like to the ones reported in figure 53. This kind of nodes are leaf nodes with only in-arcs.



hasAge	Person	Man	Woman
18..30	0.3	0.25	0.05
31..50	0.4	0.3	0.1
51..70	0.3	0.2	0.1
71..80	0.2	0.2	0

Figure 53: How to represent a data property as a LLN

We have to investigate how to make inference over the new $2lBN$, and how to extend BQ language for handling all of possible kinds of queries.

2. Dealing with logical relations between concepts introduced by OWL-DL

OWL-DL is a variation of the Ontology Web Language, and its semantics is defined based on model theroy in a way analogous to the semantics of description logics (DLs). DLs are suitable for capturing the knowledge about a domain in which instances can be grouped into classes and relationships among classes are binary. For example a class can be defined by logical operations on two or more other classes ($owl : intersectionOf$, $owl : unionOf$, $owl : complementOf$). The three logical operators correspond to AND ($conjunction$), OR ($disjunction$) and NOT ($negation$) in logic. They define classes of all individuals by standard set-operations of intersection, union, and complement, respectively. Three class axioms ($rdfs : subClassOf$, $owl : equivalentClass$, $owl : disjointWith$) can be used for defining necessary and sufficient conditions of a class. For the moment our method does not support any kind of those constraints. So, for example if we consider disjunction operator as represented in figure 54(a), the probability $P(B|C)$ tend to zero, but it is not precisely equal to zero. This problem can be solved, for example, by using the results in (DP04a), (DP04b), (DP05b), (DP05a). In this work the authors uses particular nodes, called $L - nodes$, to facilitate modeling relations among classes that are specified by OWL logical operators as reported in figure 54(b). So, we have to investigate how to integrate those kind of nodes in our $2lBN$s.

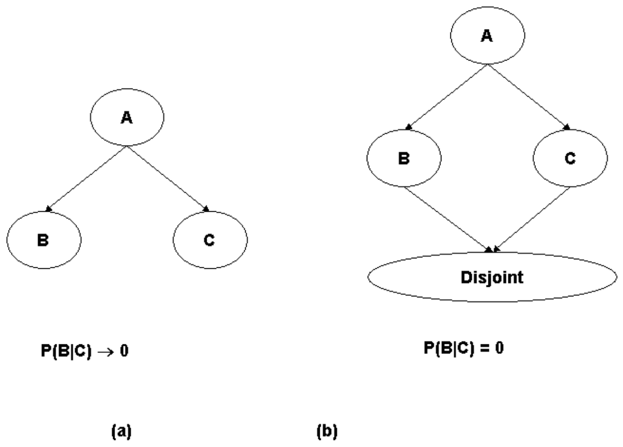


Figure 54: (a) $P(B|C)$ tends to zero. (b) $P(B|C)$ is equal to zero.

3. Dealing with cycles in $2lBN$

When the structure of the Bayesian network is not a polytree, all the reasoning recursive procedures may not terminate because there can be more than one path connecting two nodes. In particular, reflective ontology object properties can cause loops because, in that case, a node is both the father and the child of itself. For example, if we add a reflective relation $hasMother$ to Company class in figure 13, (for indicating that a Company has a mother Company) we could want to ask to the system the following question: ”*What is the probability that the CEO of a company is also a director of its mother company ?*”. The facts involved in the query are that a company c_1 has a CEO that is a person p_1, the company c_1 has a mother company that is c_2, and c_2 has a CEO that is p_1. It can be expressed by both queries and evidences conjunction as briefly described in the next Bayesian query language extensions section. In some cases recursive procedures may not terminate because $hasMother$ creates a cycle in the network. However, our main hypothesis is that the resulting graph, produced by the ontology compiling process, contains no cycles. In fact, cycles violate the properties of polytrees. Anyway, there are some proposed solutions for dealing with networks which are not polytree, and we could investigate how to integrate them in our method. For example in (Hen88), marginal probabilities of root nodes are used for assigning causal values (i.e., true, false) to those nodes. Then, by using those values, causal values are assigned to the descendents of the root nodes, by means of the conditional probability tables of those descendents, and so on till we reach the leaves. In this way, each node assumes a truth value. This process is repeated many times, remembering all the values assigned to all nodes, each time. So, we can evaluate for example $P(Q|E)$ dividing the number of times in which both Q and E have value equal to $true$ by the number of times in which E has value equal to $true$. In (Lau88) the authors group nodes of the network by $super\text{-}nodes$, in such a way that the resulting network is a polytree. This process can be repeated, grouping $super\text{-}nodes$ by other $super\text{-}nodes$, till the network is a polytree. So all the reasoning schemas for polytrees can be used for making inference, but for each $super\text{-}node$ there are many conditional probability tables. They give all the conditional probabilities for each value of each $super\text{-}node$, conditional on all the values of their father nodes, that can be $super\text{-}nodes$.

4. Bayesian query language extensions

Further investigations have to be addressed to the extension of the BQ language for handling:

1. Conjunction of evidences
2. Conjunction of queries
3. Specifying more than one conditions over the inference path

The first and the second point concern the possibility to express more evidence nodes $e_1, e_2, ...e_n$ and more query nodes $q_1, q_2, ...q_n$ respectively, in the Bayesian query. So, it could be possible to formulate queries with the following structure for the point 1:

$$P(q|_{paths}\{e_1, e_2, ..., e_n\})$$

and for the point 2:

$$P(\{q_1, q_2, ..., q_n\}|_{paths}e)$$

Point three concerns the possibility of specifying a condition along each arc composing the inference path. For the moment it is possible only to express a condition about both query and evidence node, but not also about each HLN met along the inference path.

Appendix A

Bayesian Query Language: The Operational Semantics

In the following is reported the operational semantics of the Bayesian Query Language.

Evidence both *under* **and** *over* **w.r.t. query** (1)

$$\frac{t \ll t_1, t_2 \ll t \qquad \mathcal{P}(t_2|t,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t|t_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|\bar{t},\mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|t_1,\mathcal{R}) \rightarrow n_4}{\mathcal{P}(t|t_1,t_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(1.1)

$$\frac{t \ll t_1, t_2 \ll t \qquad \mathcal{P}(t_2|t,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t|\bar{t}_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|\bar{t},\mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|\bar{t}_1,\mathcal{R}) \rightarrow n_4}{\mathcal{P}(t|\bar{t}_1,t_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

Figure 55: Taxonomy: Semantic rules about *under* and *over* evidence, part 1

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t \\ \mathcal{P}(\bar{t}_2|t,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t|t_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(\bar{t}_2|\bar{t},\mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|t_1,\mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t|t_1,\bar{t}_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.2}$$

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t \\ \mathcal{P}(\bar{t}_2|t,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t|\bar{t}_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(\bar{t}_2|\bar{t},\mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|\bar{t}_1,\mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t|\bar{t}_1,\bar{t}_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.3}$$

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t \\ \mathcal{P}(t_2|\bar{t},\mathcal{R}) \rightarrow n_1, \mathcal{P}(\bar{t}|t_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|t,\mathcal{R}) \rightarrow n_3, \mathcal{P}(t|t_1,\mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\bar{t}|t_1,t_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.4}$$

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t \\ \mathcal{P}(t_2|\bar{t},\mathcal{R}) \rightarrow n_1, \mathcal{P}(\bar{t}|\bar{t}_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|t,\mathcal{R}) \rightarrow n_3, \mathcal{P}(t|\bar{t}_1,\mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\bar{t}|\bar{t}_1,t_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.5}$$

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t \\ \mathcal{P}(\bar{t}_2|\bar{t},\mathcal{R}) \rightarrow n_1, \mathcal{P}(\bar{t}|t_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(\bar{t}_2|t,\mathcal{R}) \rightarrow n_3, \mathcal{P}(t|t_1,\mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\bar{t}|t_1,\bar{t}_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.6}$$

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t \\ \mathcal{P}(\bar{t}_2|\bar{t},\mathcal{R}) \rightarrow n_1, \mathcal{P}(\bar{t}|\bar{t}_1,\mathcal{R}) \rightarrow n_2, \mathcal{P}(\bar{t}_2|t,\mathcal{R}) \rightarrow n_3, \mathcal{P}(t|\bar{t}_1,\mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\bar{t}|\bar{t}_1,\bar{t}_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.7}$$

Evidence *over* **w.r.t. query** (2)

$$\frac{t_2 \gg t_1, t = \mathit{father}(t_1, \mathcal{R}) \quad \mathcal{P}(t_1|t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|t_2, \mathcal{R}) \rightarrow n_2}{\mathcal{P}(t_1|t_2, \mathcal{R}) \rightarrow n_1 \cdot n_2}$$

$$\frac{t_2 \gg t_1, t = \mathit{father}(t_1, \mathcal{R}) \quad \mathcal{P}(\bar{t_1}|t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|t_2, \mathcal{R}) \rightarrow n_2}{\mathcal{P}(\bar{t_1}|t_2, \mathcal{R}) \rightarrow n_1 \cdot n_2} \tag{2.1}$$

$$\frac{t_2 \gg t_1, t = \mathit{father}(t_1, \mathcal{R}) \quad \mathcal{P}(t_1|t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|\bar{t_2}, \mathcal{R}) \rightarrow n_2}{\mathcal{P}(t_1|\bar{t_2}, \mathcal{R}) \rightarrow n_1 \cdot n_2} \tag{2.2}$$

$$\frac{t_2 \gg t_1, t = \mathit{father}(t_1, \mathcal{R}) \quad \mathcal{P}(\bar{t_1}|t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|\bar{t_2}, \mathcal{R}) \rightarrow n_2}{\mathcal{P}(\bar{t_1}|\bar{t_2}, \mathcal{R}) \rightarrow n_1 \cdot n_2} \tag{2.3}$$

Figure 57: Taxonomy: Evidence *over* w.r.t. query

Evidence *under* **w.r.t. query** (3)

$$\frac{\begin{gathered} t_2 \ll t_1 \\ \mathcal{P}(t_2|t_1,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1|\varepsilon,\mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|\bar{t}_1,\mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}_1|\varepsilon,\mathcal{R}) \rightarrow n_4 \end{gathered}}{\mathcal{P}(t_1|t_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(3.1)

$$\frac{\begin{gathered} t_2 \ll t_1 \\ \mathcal{P}(t_2|\bar{t}_1,\mathcal{R}) \rightarrow n_1, \mathcal{P}(\bar{t}_1|\varepsilon,\mathcal{R}) \rightarrow n_2, \mathcal{P}(t_2|t_1,\mathcal{R}) \rightarrow n_3, \mathcal{P}(t_1|\varepsilon,\mathcal{R}) \rightarrow n_4 \end{gathered}}{\mathcal{P}(\bar{t}_1|t_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(3.2)

$$\frac{\begin{gathered} t_2 \ll t_1 \\ \mathcal{P}(\bar{t}_2|t_1,\mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1|\varepsilon,\mathcal{R}) \rightarrow n_2, \mathcal{P}(\bar{t}_2|\bar{t}_1,\mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}_1|\varepsilon,\mathcal{R}) \rightarrow n_4 \end{gathered}}{\mathcal{P}(t_1|\bar{t}_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(3.3)

$$\frac{\begin{gathered} t_2 \ll t_1 \\ \mathcal{P}(\bar{t}_2|\bar{t}_1,\mathcal{R}) \rightarrow n_1, \mathcal{P}(\bar{t}_1|\varepsilon,\mathcal{R}) \rightarrow n_2, \mathcal{P}(\bar{t}_2|t_1,\mathcal{R}) \rightarrow n_3, \mathcal{P}(\dot{t}_1|\varepsilon,\mathcal{R}) \rightarrow n_4 \end{gathered}}{\mathcal{P}(\bar{t}_1|\bar{t}_2,\mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

Figure 58: Taxonomy: Evidence *under* w.r.t. query

Evidence *over* w.r.t. query $(2bis)$

$$\frac{\begin{array}{c} t_2 \gg t_1, \{t\} = \mathit{father}(t_1, \mathcal{R}), t_2 not \in \{t\} \\ \mathcal{P}(t_1|\{t\}, \mathcal{R}) \to n_1, \exists i.t_i \in \{t\} : \mathit{ances}(t_2, t_i, \mathcal{R}), \mathcal{P}(t_i|t_2, \mathcal{R}) \to n_2 \\ \forall j : ((t_j \in \{t\}) \wedge \overline{\mathit{ances}}(t_2, t_j, \mathcal{R})).\mathcal{P}(t_j|\varepsilon, \mathcal{R}) \to n_j \end{array}}{\mathcal{P}(t_1|t_2, \mathcal{R}) \to n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t\})-1)} n_k}$$

(2bis.1)

$$\frac{\begin{array}{c} t_2 \gg t_1, \{t\} = \mathit{father}(t_1, \mathcal{R}), t_2 not \in \{t\} \\ \mathcal{P}(\bar{t_1}|\{t\}, \mathcal{R}) \to n_1, \exists i.t_i \in \{t\} : \mathit{ances}(t_2, t_i, \mathcal{R}), \mathcal{P}(t_i|t_2, \mathcal{R}) \to n_2 \\ \forall j : ((t_j \in \{t\}) \wedge \overline{\mathit{ances}}(t_2, t_j, \mathcal{R})).\mathcal{P}(t_j|\varepsilon, \mathcal{R}) \to n_j \end{array}}{\mathcal{P}(\bar{t_1}|t_2, \mathcal{R}) \to n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t\})-1)} n_k}$$

(2bis.2)

$$\frac{\begin{array}{c} t_2 \gg t_1, \{t\} = \mathit{father}(t_1, \mathcal{R}), t_2 not \in \{t\} \\ \mathcal{P}(t_1|\{t\}, \mathcal{R}) \to n_1, \exists i.t_i \in \{t\} : \mathit{ances}(t_2, t_i, \mathcal{R}), \mathcal{P}(t_i|\bar{t_2}, \mathcal{R}) \to n_2 \\ \forall j : ((t_j \in \{t\}) \wedge \overline{\mathit{ances}}(t_2, t_j, \mathcal{R})).\mathcal{P}(t_j|\varepsilon, \mathcal{R}) \to n_j \end{array}}{\mathcal{P}(t_1|\bar{t_2}, \mathcal{R}) \to n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t\})-1)} n_k}$$

(2bis.3)

$$\frac{\begin{array}{c} t_2 \gg t_1, \{t\} = \mathit{father}(t_1, \mathcal{R}), t_2 not \in \{t\} \\ \mathcal{P}(\bar{t_1}|\{t\}, \mathcal{R}) \to n_1, \exists i.t_i \in \{t\} : \mathit{ances}(t_2, t_i, \mathcal{R}), \mathcal{P}(t_i|\bar{t_2}, \mathcal{R}) \to n_2 \\ \forall j : ((t_j \in \{t\}) \wedge \overline{\mathit{ances}}(t_2, t_j, \mathcal{R})).\mathcal{P}(t_j|\varepsilon, \mathcal{R}) \to n_j \end{array}}{\mathcal{P}(\bar{t_1}|\bar{t_2}, \mathcal{R}) \to n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t\})-1)} n_k}$$

Figure 59: Taxonomy with multiply hineritance: Evidence *over* w.r.t. query

Evidence *under* **w.r.t. query** $(3bis)$

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists k. t_k = desc(t_2, t_1) \\ \mathcal{P}(t_2|t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|t_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|\bar{t}_1, \mathcal{R}) \rightarrow n_4, \\ \mathcal{P}(t_1|\varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}_1|\varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(t_1|t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{n_1 \cdot n_2 \cdot n_3 + n_1 \cdot n_4 \cdot n_5}}$$

(3bis.1)

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists k. t_k = desc(t_2, t_1) \\ \mathcal{P}(t_2|t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|\bar{t}_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|t_1, \mathcal{R}) \rightarrow n_4, \\ \mathcal{P}(\bar{t}_1|\varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t_1|\varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(\bar{t}_1|t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{n_1 \cdot n_2 \cdot n_3 + n_1 \cdot n_4 \cdot n_5}}$$

(3bis.2)

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists k. t_k = desc(t_2, t_1) \\ \mathcal{P}(\bar{t}_2|t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|t_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|\bar{t}_1, \mathcal{R}) \rightarrow n_4, \\ \mathcal{P}(t_1|\varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}_1|\varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(t_1|\bar{t}_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{n_1 \cdot n_2 \cdot n_3 + n_1 \cdot n_4 \cdot n_5}}$$

(3bis.3)

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists k. t_k = desc(t_2, t_1) \\ \mathcal{P}(\bar{t}_2|t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|\bar{t}_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|t_1, \mathcal{R}) \rightarrow n_4, \\ \mathcal{P}(\bar{t}_1|\varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t_1|\varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(\bar{t}_1|\bar{t}_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{n_1 \cdot n_2 \cdot n_3 + n_1 \cdot n_4 \cdot n_5}}$$

Figure 60: Taxonomy with multiply hineritance: Evidence *under* w.r.t. query

Evidence *under* w.r.t. query (query is a father of evidence) (3.1)

$$\frac{t_2 \ll t_1, t_1 = \mathit{father}(t_2) \quad \mathcal{P}(t_2|t_1,\mathcal{R}) \to n_1, \mathcal{P}(t_1,\mathcal{R}) \to n_2, \mathcal{P}(t_2|\bar{t}_1,\mathcal{R}) \to n_3, \mathcal{P}(\bar{t}_1|\varepsilon,\mathcal{R}) \to n_4}{\mathcal{P}(t_1|t_2,\mathcal{R}) \to \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(3.1.1)

$$\frac{t_2 \ll t_1, t_1 = \mathit{father}(t_2) \quad \mathcal{P}(t_2|\bar{t}_1,\mathcal{R}) \to n_1, \mathcal{P}(\bar{t}_1,\mathcal{R}) \to n_2, \mathcal{P}(t_2|t_1,\mathcal{R}) \to n_3, \mathcal{P}(t_1|\varepsilon,\mathcal{R}) \to n_4}{\mathcal{P}(\bar{t}_1|t_2,\mathcal{R}) \to \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(3.1.2)

$$\frac{t_2 \ll t_1, t_1 = \mathit{father}(t_2) \quad \mathcal{P}(\bar{t}_2|t_1,\mathcal{R}) \to n_1, \mathcal{P}(t_1,\mathcal{R}) \to n_2, \mathcal{P}(\bar{t}_2|\bar{t}_1,\mathcal{R}) \to n_3, \mathcal{P}(\bar{t}_1|\varepsilon,\mathcal{R}) \to n_4}{\mathcal{P}(t_1|\bar{t}_2,\mathcal{R}) \to \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(3.1.3)

$$\frac{t_2 \ll t_1, t_1 = \mathit{father}(t_2) \quad \mathcal{P}(\bar{t}_2|\bar{t}_1,\mathcal{R}) \to n_1, \mathcal{P}(\bar{t}_1,\mathcal{R}) \to n_2, \mathcal{P}(\bar{t}_2|t_1,\mathcal{R}) \to n_3, \mathcal{P}(t_1|\varepsilon,\mathcal{R}) \to n_4}{\mathcal{P}(\bar{t}_1|\bar{t}_2,\mathcal{R}) \to \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

Figure 61: Taxonomy with multiply hineritance: query is one of the fathers of the evidence

Probability distribution of a node given its $\mathcal{C}(\{t\})$ fathers (14)

$$\frac{\begin{array}{c} n_1 = \#_{inst}((\Phi^{-1}(t) \cap (\bigcap_{i=1}^{\mathcal{C}(\{t\})} \Phi^{-1}(t_i))), \mathcal{R}) \\ n_2 = \#_{inst}(\bigcap_{i=1}^{\mathcal{C}(\{t\})} \Phi^{-1}(t_i), \mathcal{R}) \end{array}}{\mathcal{P}(t|\{t\}, \mathcal{R}) \to \frac{n_1}{n_2}}$$

(14.1)

$$\frac{\begin{array}{c} n_1 = \#_{inst}((\Phi^{-1}(\bar{t}) \cap (\bigcap_{i=1}^{\mathcal{C}(\{t\})} \Phi^{-1}(t_i))), \mathcal{R}) \\ n_2 = \#_{inst}(\bigcap_{i=1}^{\mathcal{C}(\{t\})} \Phi^{-1}(t_i), \mathcal{R}) \end{array}}{\mathcal{P}(\bar{t}|\{t\}, \mathcal{R}) \to \frac{n_1}{n_2}}$$

Figure 62: Taxonomy with multiply hineritance: probability distribution of a node given its fathers

Evidence is one of the fathers of the query (15)

$$\frac{\{t\} = \mathit{father}(t_1, \mathcal{R}), t_2 \in \{t\}, \mathcal{C}(\{t\}) \geq 2 \quad \forall i.t_i \in \{\{t\}/t_2\} : \mathcal{P}(t_i|\varepsilon, \mathcal{R}) \to n_i \quad \mathcal{P}(t_1|\{t\}, \mathcal{R}) \to n_1}{\mathcal{P}(t_1|t_2, \mathcal{R}) \to n_1 \cdot \prod_{i=1}^{(\mathcal{C}(\{t\})-1)} n_i}$$

$$\frac{\{t\} = \mathit{father}(t_1, \mathcal{R}), t_2 \in \{t\}, \mathcal{C}(\{t\}) \geq 2 \quad \forall i.t_i \in \{\{t\}/t_2\} : \mathcal{P}(t_i|\varepsilon, \mathcal{R}) \to n_i \quad \mathcal{P}(\overline{t}_1|\{t\}, \mathcal{R}) \to n_1}{\mathcal{P}(\overline{t}_1|t_2, \mathcal{R}) \to n_1 \cdot \prod_{i=1}^{(\mathcal{C}(\{t\})-1)} n_i} \tag{15.1}$$

$$\frac{\{t\} = \mathit{father}(t_1, \mathcal{R}), \overline{t}_2 \in \{t\}, \mathcal{C}(\{t\}) \geq 2 \quad \forall i.t_i \in \{\{t\}/t_2\} : \mathcal{P}(t_i|\varepsilon, \mathcal{R}) \to n_i \quad \mathcal{P}(t_1|\{t\}, \mathcal{R}) \to n_1}{\mathcal{P}(t_1|\overline{t}_2, \mathcal{R}) \to n_1 \cdot \prod_{i=1}^{(\mathcal{C}(\{t\})-1)} n_i} \tag{15.2}$$

$$\frac{\{t\} = \mathit{father}(t_1, \mathcal{R}), \overline{t}_2 \in \{t\}, \mathcal{C}(\{t\}) \geq 2 \quad \forall i.t_i \in \{\{t\}/t_2\} : \mathcal{P}(t_i|\varepsilon, \mathcal{R}) \to n_i \quad \mathcal{P}(\overline{t}_1|\{t\}, \mathcal{R}) \to n_1}{\mathcal{P}(\overline{t}_1|\overline{t}_2, \mathcal{R}) \to n_1 \cdot \prod_{i=1}^{(\mathcal{C}(\{t\})-1)} n_i} \tag{15.3}$$

Figure 63: Taxonomy with multiply hineritance: evidence is one of the father of the query

Evidence both *under* and *over* w.r.t. query (1)

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(t_2|_{p_2} t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|_{p_1} t_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(t_2|_{p_2} \bar{t}, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|_{p_1} t_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t|_p(t_1, t_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.1}$$

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(t_2|_{p_2} t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|_{p_1} \bar{t}_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(t_2|_{p_2} \bar{t}, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|_{p_1} \bar{t}_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t|_p(\bar{t}_1, t_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.2}$$

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(\bar{t}_2|_{p_2} t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|_{p_1} t_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(\bar{t}_2|_{p_2} \bar{t}, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|_{p_1} t_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t|_p(t_1, \bar{t}_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}} \tag{1.3}$$

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(\bar{t}_2|_{p_2} t, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t|_{p_1} \bar{t}_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(\bar{t}_2|_{p_2} \bar{t}, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}|_{p_1} \bar{t}_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t|_p(\bar{t}_1, \bar{t}_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

Figure 64: Polytree: Evidence both *over* and *under* w.r.t. query, part 1

(1.4)

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(t_2|_{p_2}\overline{t}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(\overline{t}|_{p_1}t_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(t_2|_{p_2}t, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t|_{p_1}t_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\overline{t}|_p(t_1, t_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(1.5)

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(t_2|_{p_2}\overline{t}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(\overline{t}|_{p_1}\overline{t}_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(t_2|_{p_2}t, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t|_{p_1}\overline{t}_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\overline{t}|_p(\overline{t}_1, t_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(1.6)

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(\overline{t}_2|_{p_2}\overline{t}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(\overline{t}|_{p_1}t_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(\overline{t}_2|_{p_2}t, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t|_{p_1}t_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\overline{t}|_p(t_1, \overline{t}_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

(1.7)

$$\frac{\begin{array}{c} t \ll t_1, t_2 \ll t, \exists p_1, p_2 : (t_1.p_1 \Rightarrow^* t) \wedge (t.p_2 \Rightarrow^* t_2) \wedge (p = p_1.p_2) \\ \mathcal{P}(\overline{t}_2|_{p_2}\overline{t}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(\overline{t}|_{p_1}\overline{t}_1, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(\overline{t}_2|_{p_2}t, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t|_{p_1}\overline{t}_1, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\overline{t}|_p(\overline{t}_1, \overline{t}_2), \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{n_1 \cdot n_2 + n_3 \cdot n_4}}$$

Evidence *over* **w.r.t. query** (2)

$$\frac{\begin{array}{c} t_2 \gg t_1, \exists p : (t_2.p \Rightarrow^* t_1) \\ \{t_0\} = \mathit{father}(t_1, \mathcal{R}), \{p_0\} = \mathit{arc}(t_1, \{t_0\}, \mathcal{R}), \exists j : t_j.\mathit{tail}(p) \Rightarrow^* t_2 \\ \mathcal{P}(t_1|_{\{p_0\}}\{t_0\}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_j|_{\mathit{tail}(p)} t_2, \mathcal{R}) \rightarrow n_2 \\ \forall i : ((t_i \in \{t_0\}) \wedge \overline{\mathit{ances}}(t_2, t_i, \mathcal{R}) \wedge (p_i = \mathit{arc}(t_1, t_i, \mathcal{R})).\mathcal{P}(t_i|_{p_i}\varepsilon, \mathcal{R}) \end{array}}{\mathcal{P}(t_1|_p t_2, \mathcal{R}) \rightarrow n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t_0\})-1)} n_k}$$

(2.1)

$$\frac{\begin{array}{c} t_2 \gg t_1, \exists p : (t_2.p \Rightarrow^* t_1) \\ \{t_0\} = \mathit{father}(t_1, \mathcal{R}), \{p_0\} = \mathit{arc}(t_1, \{t_0\}, \mathcal{R}), \exists j : t_j.\mathit{tail}(p) \Rightarrow^* t_2 \\ \mathcal{P}(\bar{t}_1|_{\{p_0\}}\{t_0\}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_j|_{\mathit{tail}(p)} t_2, \mathcal{R}) \rightarrow n_2 \\ \forall i : ((t_i \in \{t_0\}) \wedge \overline{\mathit{ances}}(t_2, t_i, \mathcal{R}) \wedge (p_i = \mathit{arc}(t_1, t_i, \mathcal{R})).\mathcal{P}(t_i|_{p_i}\varepsilon, \mathcal{R}) \end{array}}{\mathcal{P}(\bar{t}_1|_p t_2, \mathcal{R}) \rightarrow n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t_0\})-1)} n_k}$$

(2.2)

$$\frac{\begin{array}{c} t_2 \gg t_1, \exists p : (t_2.p \Rightarrow^* t_1) \\ \{t_0\} = \mathit{father}(t_1, \mathcal{R}), \{p_0\} = \mathit{arc}(t_1, \{t_0\}, \mathcal{R}), \exists j : t_j.\mathit{tail}(p) \Rightarrow^* t_2 \\ \mathcal{P}(t_1|_{\{p_0\}}\{t_0\}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_j|_{\mathit{tail}(p)} \bar{t}_2, \mathcal{R}) \rightarrow n_2 \\ \forall i : ((t_i \in \{t_0\}) \wedge \overline{\mathit{ances}}(t_2, t_i, \mathcal{R}) \wedge (p_i = \mathit{arc}(t_1, t_i, \mathcal{R})).\mathcal{P}(t_i|_{p_i}\varepsilon, \mathcal{R}) \end{array}}{\mathcal{P}(t_1|_p \bar{t}_2, \mathcal{R}) \rightarrow n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t_0\})-1)} n_k}$$

(2.3)

$$\frac{\begin{array}{c} t_2 \gg t_1, \exists p : (t_2.p \Rightarrow^* t_1) \\ \{t_0\} = \mathit{father}(t_1, \mathcal{R}), \{p_0\} = \mathit{arc}(t_1, \{t_0\}, \mathcal{R}), \exists j : t_j.\mathit{tail}(p) \Rightarrow^* t_2 \\ \mathcal{P}(\bar{t}_1|_{\{p_0\}}\{t_0\}, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_j|_{\mathit{tail}(p)} \bar{t}_2, \mathcal{R}) \rightarrow n_2 \\ \forall i : ((t_i \in \{t_0\}) \wedge \overline{\mathit{ances}}(t_2, t_i, \mathcal{R}) \wedge (p_i = \mathit{arc}(t_1, t_i, \mathcal{R})).\mathcal{P}(t_i|_{p_i}\varepsilon, \mathcal{R}) \end{array}}{\mathcal{P}(\bar{t}_1|_p \bar{t}_2, \mathcal{R}) \rightarrow n_1 \cdot n_2 \cdot \prod_{k=1}^{(\mathcal{C}(\{t_0\})-1)} n_k}$$

Figure 65: Polytree: Evidence *over* w.r.t. query

Evidence *under* **w.r.t. query** (3)

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}) \\ \exists k : (1 \leq k \leq \mathcal{C}(\{t_0\}))t_1.\mathit{head}(p) \Rightarrow^* t_k, p_l = \mathit{arc}(t_2, t_k, \mathcal{R}) \\ \mathcal{P}(t_2|_{p_l} t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|_{\mathit{head}(p)} t_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|_{\mathit{head}(p)} \overline{t}_1, \mathcal{R}) \rightarrow n_4 \\ \exists p_m : p = p_m.\mathit{tail}(p), \mathcal{P}(t_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\overline{t}_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(t_1|_p t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{(n_1 \cdot n_2 \cdot n_3) + (n_1 \cdot n_4 \cdot n_5)}}$$

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}) \\ \exists k : (1 \leq k \leq \mathcal{C}(\{t_0\}))t_1.\mathit{head}(p) \Rightarrow^* t_k, p_l = \mathit{arc}(t_2, t_k, \mathcal{R}) \\ \mathcal{P}(t_2|_{p_l} t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|_{\mathit{head}(p)} \overline{t}_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|_{\mathit{head}(p)} t_1, \mathcal{R}) \rightarrow n_4 \\ \exists p_m : p = p_m.\mathit{tail}(p), \mathcal{P}(\overline{t}_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(\overline{t}_1|_p t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{(n_1 \cdot n_2 \cdot n_3) + (n_1 \cdot n_4 \cdot n_5)}} \tag{3.1}$$

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}) \\ \exists k : (1 \leq k \leq \mathcal{C}(\{t_0\}))t_1.\mathit{head}(p) \Rightarrow^* t_k, p_l = \mathit{arc}(t_2, t_k, \mathcal{R}) \\ \mathcal{P}(\overline{t}_2|_{p_l} t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|_{\mathit{head}(p)} t_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|_{\mathit{head}(p)} \overline{t}_1, \mathcal{R}) \rightarrow n_4 \\ \exists p_m : p = p_m.\mathit{tail}(p), \mathcal{P}(t_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\overline{t}_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(t_1|_p \overline{t}_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{(n_1 \cdot n_2 \cdot n_3) + (n_1 \cdot n_4 \cdot n_5)}} \tag{3.2}$$

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}) \\ \exists k : (1 \leq k \leq \mathcal{C}(\{t_0\}))t_1.\mathit{head}(p) \Rightarrow^* t_k, p_l = \mathit{arc}(t_2, t_k, \mathcal{R}) \\ \mathcal{P}(\overline{t}_2|_{p_l} t_k, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_k|_{\mathit{head}(p)} \overline{t}_1, \mathcal{R}) \rightarrow n_2, \mathcal{P}(t_k|_{\mathit{head}(p)} t_1, \mathcal{R}) \rightarrow n_4 \\ \exists p_m : p = p_m.\mathit{tail}(p), \mathcal{P}(\overline{t}_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t_1|_{p_m} \varepsilon, \mathcal{R}) \rightarrow n_5 \end{array}}{\mathcal{P}(\overline{t}_1|_p \overline{t}_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2 \cdot n_3}{(n_1 \cdot n_2 \cdot n_3) + (n_1 \cdot n_4 \cdot n_5)}} \tag{3.3}$$

Figure 66: Polytree: Evidence *under* w.r.t. query

Evidence $under$ w.r.t. query (query is a father of evidence) (3.1)

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}), t_1 \in \{t_0\} \\ p_l = \mathit{arc}(t_2, t_1, \mathcal{R}), \mathcal{P}(t_2|_{p_l} t_1, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(t_2|_{p_l} \bar{t}_1, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t_1|_p t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{(n_1 \cdot n_2) + (n_3 \cdot n_4)}}$$

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}), t_1 \in \{t_0\} \\ p_l = \mathit{arc}(t_2, t_1, \mathcal{R}), \mathcal{P}(t_2|_{p_l} \bar{t}_1, \mathcal{R}) \rightarrow n_1, \mathcal{P}(\bar{t}_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(t_2|_{p_l} t_1, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\bar{t}_1|_p t_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{(n_1 \cdot n_2) + (n_3 \cdot n_4)}} \quad (3.1.1)$$

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}), t_1 \in \{t_0\} \\ p_l = \mathit{arc}(t_2, t_1, \mathcal{R}), \mathcal{P}(\bar{t}_2|_{p_l} t_1, \mathcal{R}) \rightarrow n_1, \mathcal{P}(t_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(\bar{t}_2|_{p_l} \bar{t}_1, \mathcal{R}) \rightarrow n_3, \mathcal{P}(\bar{t}_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(t_1|_p \bar{t}_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{(n_1 \cdot n_2) + (n_3 \cdot n_4)}} \quad (3.1.2)$$

$$\frac{\begin{array}{c} t_2 \ll t_1, \exists p : (t_1.p \Rightarrow^* t_2) \\ \{t_0\} = \mathit{father}(t_2, \mathcal{R}), t_1 \in \{t_0\} \\ p_l = \mathit{arc}(t_2, t_1, \mathcal{R}), \mathcal{P}(\bar{t}_2|_{p_l} \bar{t}_1, \mathcal{R}) \rightarrow n_1, \mathcal{P}(\bar{t}_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_2 \\ \mathcal{P}(\bar{t}_2|_{p_l} t_1, \mathcal{R}) \rightarrow n_3, \mathcal{P}(t_1|_{p_l} \varepsilon, \mathcal{R}) \rightarrow n_4 \end{array}}{\mathcal{P}(\bar{t}_1|_p \bar{t}_2, \mathcal{R}) \rightarrow \frac{n_1 \cdot n_2}{(n_1 \cdot n_2) + (n_3 \cdot n_4)}} \quad (3.1.3)$$

Figure 67: Polytree: query is one of the fathers of the evidence

Probability distribution of a node given its $\mathcal{C}(\{t\})$ fathers (4)

$$\frac{\begin{array}{c}\{t\} = \mathit{father}(t,\mathcal{R}), \mathcal{S} = \mathcal{IC}(t,\mathcal{R}) \\ \forall i : i = 1,...,\mathcal{C}(\{t\}).((t_i \in \{t\}) \wedge (p_i \in \{p\}) \wedge (p_i = \mathit{arc}(t,t_i,\mathcal{R}))) \\ n_2 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), \mathit{upper}(\Phi^{-1}(t))) \\ n_1 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), (\Phi^{-1}(t)) \models \mathcal{S})\end{array}}{\mathcal{P}(t|_p\{t\},\mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

(4.1)

$$\frac{\begin{array}{c}\{t\} = \mathit{father}(t,\mathcal{R}), \mathcal{S} = \mathcal{IC}(t,\mathcal{R}) \\ \forall i : i = 1,...,\mathcal{C}(\{t\}).((t_i \in \{t\}) \wedge (p_i \in \{p\}) \wedge (p_i = \mathit{arc}(t,t_i,\mathcal{R}))) \\ n_2 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), \mathit{upper}(\Phi^{-1}(\overline{t}))) \\ n_1 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), (\Phi^{-1}(\overline{t})) \models \mathcal{S})\end{array}}{\mathcal{P}(\overline{t}|_p\{t\},\mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

Figure 68: Polytree: probability distribution of a node given its fathers

Probability distribution reversed (5)

$$\frac{\begin{array}{c}\{t\} = father(t,\mathcal{R})\\ \forall i : i = 1,\ldots,\mathcal{C}(\{t\}).((t_i \in \{t\}) \wedge (p_i \in \{p\}) \wedge (p_i = arc(t,t_i,\mathcal{R})))\\ \exists i.(t_i \in \{t\}) \wedge (\mathcal{IC}(t_i) = \mathcal{S})\\ n_1 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), (\Phi^{-1}(t)))\\ n_2 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), upper(\Phi^{-1}(t)))\\ \text{where } \exists k.(t_k \in \{t\}) \wedge ((\Phi^{-1}(t_k)) \models \mathcal{S})\end{array}}{\mathcal{P}(t|_p\{t\},\mathcal{R}) \to \frac{n_1}{n_2}}$$

(5.1)

$$\frac{\begin{array}{c}\{t\} = father(t,\mathcal{R})\\ \forall i : i = 1,\ldots,\mathcal{C}(\{t\}).((t_i \in \{t\}) \wedge (p_i \in \{p\}) \wedge (p_i = arc(t,t_i,\mathcal{R})))\\ \exists i.(t_i \in \{t\}) \wedge (\mathcal{IC}(t_i) = \mathcal{S})\\ n_1 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), (\Phi^{-1}(\overline{t})))\\ n_2 = \#_{triple}(\Phi^{-1}(\{t_i\}), \Psi^{-1}(\{p_i\}), upper(\Phi^{-1}(\overline{t})))\\ \text{where } \exists k.(t_k \in \{t\}) \wedge ((\Phi^{-1}(t_k)) \models \mathcal{S})\end{array}}{\mathcal{P}(\overline{t}|_p\{t\},\mathcal{R}) \to \frac{n_1}{n_2}}$$

Figure 69: Polytree: Reversed probability distribution

A priori induced probability distribution (6)

$$\frac{\begin{array}{c}\mathcal{IC}(t) = \mathcal{S}\\ n_1 = \#_{inst}(\Phi^{-1}(t) \models \mathcal{S})\\ n_2 = \#_{inst}(upper(\Phi^{-1}(t)))\end{array}}{\mathcal{P}(t|\varepsilon,\mathcal{R}) \to \frac{n_1}{n_2}}$$

(6.1)

$$\frac{\begin{array}{c}\mathcal{IC}(t) = \mathcal{S}\\ n_1 = \#_{inst}(\Phi^{-1}(\overline{t}) \models \mathcal{S})\\ n_2 = \#_{inst}(upper(\Phi^{-1}(\overline{t})))\end{array}}{\mathcal{P}(\overline{t}|\varepsilon,\mathcal{R}) \to \frac{n_1}{n_2}}$$

Figure 70: Polytree: A prior induced probability distribution

A priori probability distribution (7)

$$\frac{\begin{array}{c} root(t,\mathcal{R}), \exists t_1.(t.p \Rightarrow t_1) \\ n_1 = \#_{triple}(upper(\Phi^{-1}(t)), \Psi^{-1}(p), upper(\Phi^{-1}(t_1))) \\ n_2 = \#_{inst}(upper(\Phi^{-1}(t))) \end{array}}{\mathcal{P}(t|_p\varepsilon, \mathcal{R}) \rightarrow \frac{n_1}{n_2}}$$

$$\frac{\begin{array}{c} root(t,\mathcal{R}), \exists t_1.(t.p \Rightarrow t_1) \\ n_1 = \#_{triple}(upper(\Phi^{-1}(\overline{t})), \Psi^{-1}(p), upper(\Phi^{-1}(t_1))) \\ n_2 = \#_{inst}(upper(\Phi^{-1}(\overline{t}))) \end{array}}{\mathcal{P}(\overline{t}|_p\varepsilon, \mathcal{R}) \rightarrow \frac{n_1}{n_2}} \tag{7.1}$$

Figure 71: Polytree: A prior probability distribution

References

[ABC] The Internet Classic Archive. Aristotele (350 B.C.) Categories. 9

[AH05] S.; Agarwal and P. Hitzler. Modelling Fuzzy Rules with Description Logics. *In Proceedings of Workshop on OWL Experiences and Directions*, 2005. 39

[Bac90] F. Bacchus. Representing and Reasoning with Probabilistic Knowledge. *Cambridge, MA: MIT Press*, 1990. 38

[BP03] D.; McGuinness D.; Nardi D. Baader, F.; Calvanese and P. PatelSchneider. The Description Logic handbook. *Cambridge University Press.*, 2003. 13

[Cha99] J. R.; Benjamins V. R. Chandrasekaran, B.; Josephson. What Are Ontologies and Why Do We Need Them. *IEEE Intelligence Systems*, 1999. 1

[CK05] K. B.; Costa, P. C. G.; Laskey and Laskey K.J. PR-OWL: A Bayesian Ontology Language for the Semantic Web. *In Proceedings of ISWC-URSW*, pages 25–33, 2005. 2, 36, 37, 39

[Coo90] G. F. Cooper. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence*, 42:393–405, 1990. 33

[Cos08] M.; Carvahlo R. N.; Laskey K. B.; Santos L. L.; Matsumoto S. Costa, P. C. G.; Ladeira. A First-Order Bayesian Tool for Probabilistic Ontologies. *In Proceedings of 21st Florida FLAIRS UNBBayes-MEBN*, 2008. 37

[Dag93] M. Dagum, P.; Luby. Approximating Probabilistic Inference in Bayesian Belief Networks is NP-Hard. *Artificial Intelligence*, 60:141–153, 1993. 33

[DP04a] Y.; Ding, Z.; Peng and R. Pan. A Bayesian Approach to Uncertainty Modelling in OWL Ontology. *In Proceedings of 2004 International Conference on Advances in Intelligent Systems*, 2004. 35, 36, 114

[DP04b] Z.; Ding and Y. Peng. A Probabilistic Extension to Ontology Language OWL. *In Proceedings of the 37th Hawaii International Conference on system Sciences*, 2004. 35, 36, 114

[DP05a] Y.; Ding, Z.; Peng and R. Pan. BayesOWL: Uncertainty Modeling in Semantic Web Ontologies. *In Soft Computing in Ontologies and Semantic Web, Springer-Verlag*, 2005. 35, 36, 39, 114

[DP05b] Z.; Ding and Y. Peng. Modifying Bayesian Networks by Probabilistic Constraints. *In Proceedings of UAI-2005, Edimburgh, Scotland*, 2005. 35, 36, 114

[Fla00] N. Flach, P. A.; Lachiche. Decomposing Probability Distributions on Structured Individuals. *Work-in-Progress Reports of the 10th International Conference on Inductive Logic Programming*, pages 96–106, 2000. 40

[Fry89] M. Frydemberg. The Chain Graph Markov Property. *Scandinavian Journal of Statistics*, 17:333–353, 1989. 17

[Get02] L. C. Getoor. Learning Statistical Models from Relational Data. *Ph.D. dissertation, Department of Computer Science, Stanford University*, 2002. 38

[GL02] R.; Giugno and T. Lukasiewicz. P-SHOQ(D): A Probabilistic Extension of SHOQ(D) for Probabilistic Ontologies in the Semantic Web. *INFSYS, Research Report*, 2002. 39

[GN87] M. R. Genesereth and N. Nilsson. Logical Foundations of Artificial Intelligence. *Morgan Kaufmann Publishers: San Mateo, CA*, 1987. 11

[Gyf04] P. A. Gyftodimos, E.; Flach. Hierarchical Bayesian Network An Approach to Classification and Learning from Structured Data. *Knowledge Representation and Search*, pages 291–300, 2004. 40

[H.67] Melay G. H. Another look at data. *In Proceedings of the Fall Joint Computer Conference*, 31:525–534, 1967. 10

[Hal90] J. Y. Halpern. An Analisys of first-order Logics of Probability. *Artificial Intelligence*, 46:311–350, 1990. 38

[Hei94] J. Heinsohn. Probabilistic Description Logics. *In Proceedings of UAI*, pages 311–318, 1994. 39

[Hen88] M. Henrion. Propagation of Uncertainty in Bayesian Networks by Probabilistic Logic Sampling. *in Lemmer, J, e Kanal, L. Uncertainty in Artificial Intelligence*, 2:149–163, 1988. 115

[HH04] M.; Holi and E. Hyvonen. Probabilistic Information retrieval Based on Conceptual Overlap in Semantic Web Ontologies. *In Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, 2004. 38

[HH05] M.; Holi and E. Hyvonen. Modelling Degrees of Conceptual Overlap in Semantic Web Ontologies. *In Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, 2005. 38

[Hor04] I. Horrocks. SWRL: A Semantic Web Rule language Combining OWL and RuleML. *http://www.w3c.org/Submission/2004/SUBM-SWRL-20040521*, 2004. 15

[HS05] H. I.; Haarsler, V.; Pai and N. Shiri. A generic Framework for Description Logics with Uncertainty. *In Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, 2005. 39

[Jae94] M. Jaeger. Probabilistic Reasoning in Terminological Logics. *In Proceedings of KR*, pages 305–316, 1994. 39

[Kad96] D. A. Kadane, J. B.; Schum. A Probabilistic Analysis of the Scacco and Vanzetti Evidence. 1996. 3

[Kim83] J. Kim, J. H.; Pearl. A Computational Model for Causal and Diagnostic Reasoning in Inference systems. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 190–193, 1983. 34

[Kol97] A.; Pfeffer A. Koller, D.; Levy. P-CLASSIC A Tractable Probabilistic Description Logic. *In Proceedings of AAAI*, pages 390–397, 1997. 38

[Las05] P. C. G. Laskey, K. B.; Costa. Bayesian logic for the 23rd Century. *In Proceedings of Uncertainty in Artificial Intelligence*, 2005. 36

[Las07] K. B. Laskey. MEBN: A Language for First-Order Bayesian Knowledge Bases. *In Proceedings of Artificial Intelligence*, 2007. 2

[Lau88] D. Lauritzen, S.; Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert System. *Journal of the Royal Statistical Society B 50(2)*, pages 157–224, 1988. 115

[Lit05] G. Little, E.; Rogova. Ontology Meta-Model for Building A Situational Picture of Catastrophic Events. *In Proceedings of the FUSION 2005-8th International Conference on Multisource Information Fusion*, 2005. 4

[Lit06a] E. Little. The Need for Metaphysically-based Ontologies in Higher-level Information Fusion Applications. *Contribution to the Third International Workshop on Philosophy and Informatics*, 2006. 3

[Lit06b] G. Little, E.; Rogova. Principles for The Development of Upper Ontologies in Higher-Level Information Fusion Applications. *In Proceedings of the 2006 Formal Ontology for Information Systems Conference*, 2006. 4

[LJTP] Basilia (1613). Lorhard J. Theatrum Philosophicum. 9

[MD05] M.; Mazzieri and A. F. Dragoni. A Fuzzy Semantics for Semantic Web Languages. *In Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, 2005. 39

[NF04] H.; Nottelmann and N. Fuhr. A Probabilistic Extension to DAML+OIL based on Probabilistic Datalog. *In Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based systems (IPMU)*, 2004. 39

[PA05] F.; Cannon S.; Pool, M.; Fung and J. Aikin. Is it Worth a Hoot? Qualms about OWL for Uncertainty Reasoning. *In Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, 2005. 38

[Pea88] J. Pearl. Probabilistic Reasoning in Intelligence Systems: Networks of Plausible Inference. *San Mateo, CA: MOrgan Kaufmann*, 1988. 3, 22, 34

[PH05] G.; Tzouvaras V.; Pan, J. Z.; Stamou and I. Horrocks. f-SWRL: A Fuzzy Extension of SWRL. *In Proceedings of the International Conference on Artificial Neural Networks*, 2005. 39

[Poo93] D. Poole. Probabilistic Horn Abduction and Bayesian Networks. *Artificial Intelligence*, 64-1:81–129, 1993. 39

[RM03] A. A. Razmerita, L.; Angehrn and A. Maedche. Ontology-Based User Modeling for Knowledge Management Systems. *User Modeling*, pages 213–217, 2003. 10

[Sch94] D. A. Schum. Evidential Foundations of Probabilistic Reasoning. *New York, NY, USA: Wiley*, 1994. 3

[Sha86] G. Shafer. the Construction of Probability Arguments. *Boston University Law Review*, (66(3-4)):799–816, 1986. 3

[Smi95] D. W. Smith, B.; Smith. The Cambridge Companion to Husserl - Cambridge: Cambridge University Press. pages 27–29, 1995. 3

[Sol] MUSING: Multi-Industri Semantic-Based Business Intelligence Solutions. 69

[Sto05] G.; Tzouvaras V.; Pan J.; Horrocks I. Stoilos, G.; Stamou. The Fuzzy Description Logic f-SHIN. *In Proceedings of Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, 2005. 39

[Str04] U. Straccia. Uncertainty and Description Logic Programs: A Proposal for Expressing Rules and Uncertainty on top of Ontologies. *Technical Report, CNR Pisa ISTI-2004-TR*, 2004. 39

[Str05] U. Straccia. A Fuzzy Description Logic for the Semantic Web. *In Capturing Intelligenece: Fuzzy Logic and the Semantic Web, E. Sanchez, Ed. Elsevier*, 2005. 39

[Wer90] S. L. Wermuth, N.; Lauritzen. On Substantive Research Hypotheses, Conditional Independence Graphs and Graphical Chain Models. *Journal of The Royal Statistical Society*, 52:21–50, 1990. 17

[Wri06] K. B. Wright, E.; Laskey. Credibility Models for Multi-Source Fusion. *In Proceedings of the Ninth International Conference on Information Fusion*, 2006. 4

[Yel99] P. M. Yelland. Market Analysis using Combination of Bayesian Networks and Description Logics. *Sun Microsystem, Technical Report*, TR-99-78, 1999. 38