A two-way approach for probabilistic graphical models structure learning and ontology enrichment

by:

Mouna Ben Ishak

supervisors:

Dr. Nahla Ben Amor (ISG, Tunisie)

Pr. Philippe Leray (Polytech'Nantes, France)

To my dear parents,
to my dear brothers,
to all my family members,
to all those who love me.

Acknowledgements

First, I thank the jury's members who honored me by judging my dissertation. I am grateful for the time they spent to do this.

Then, I want to express my deep gratitude and sincere appreciation to my two supervisors: Dr. Nahla Ben Amor who often gave me the courage to move forward in my research, I am thankful for her continuous support, her precious advices and her invaluable help, Pr. Philippe Leray for the fruitful discussions I had with him and for his kindness and unmatched modesty, I was fortunate to learn from him and to benefit from his expertise in this interesting research area. Without them this dissertation would never have been achieved.

Finally, an honorable mention goes to my dear family. I would like to offer my heartiest thanks to my parents for their understanding, support and confidence to which I am forever indebted and to my brothers for listening to my complaints and frustrations and for their continuous help. Words are not enough to express how much I respect and I love them and I am sure that having brought this work forward is the best reward for them.

Let all those who helped me by their support and encouragement find here the expression of my sincere gratitude.

Contents

List of Figures

Introduction

Knowledge representation (KR) is one of the principal areas of Artificial Intelligence which was studied by different techniques coming from various disciplines. In this work we will focus on probabilistic graphical models and ontologies which are considered within the most efficient frameworks in KR.

Derived from both graph and probability theories, probabilistic Graphical Models (Koller and Friedman, 2009) (PGMs) are powerful tools for representing and reasoning under uncertainty. Although useful in several domains, they suffer from their building phase known to be mostly an NP-hard problem which can limit in some extent their application, especially in real world. Bayesian networks (BNs) (Pearl, 1988) are among the most used PGMs. Despite their great success, BNs are limited when dealing with large-scale systems. Thus, several extensions have been proposed in order to broaden their range of application, such as object oriented Bayesian networks (OOBNs) (Bangsø and Wuillemin, 2000; Bangsø et al., 2000; Koller and Pfeffer, 1997) which introduce the object paradigm into the framework of BNs.

Ontologies (Gruber, 1993) are another tool for knowledge representation whose use is increasingly prevalent by the computer science community. They provide a structured representation of knowledge characterized by its semantic richness and represent the basis for deductive reasoning. It is well known that the ontology enrichment is a very hard problem based mainly on human expertise, which makes it a hard and error-prone process given the large and complex structures of ontologies. Therefore, several algorithms were proposed to automate this process in order to improve both its speed and reliability.

Even though they represent two different paradigms, PGMs and ontologies share several similarities which has led to some research directions aiming to combine them. The concern for the majority of them was to extend ontologies in order to support uncertainty. This is either by adding additional markups to represent probabilistic information or by mapping the ontology into a PGM in order to en-

rich ontology reasoning with probabilistic queries. Few works intend to construct PGMs using ontologies. In this area, Bayesian networks are the most commonly used. Typically, concepts are associated to nodes, ontology relations are used to link these nodes, and for some proposals, axioms are involved to express nodes or edges or to define the states of variables. However, given the restrictive expressiveness of Bayesian networks, these methods focus on a restrained range of ontologies and neglect some of their components such as representing concepts properties, non taxonomic relations, etc.

In this work, we propose to combine these two tools in a two-way approach that allows PGMs and ontologies cooperation. Our idea is to harness ontologies representation capabilities in order to enrich the building process of PGMs. Then, to use the result of the learning process in order to enrich the ontology used initially. We propose to use advanced PGMs in order to overcome standard BNs weaknesses and handle different ontologies without significant restrictions. We are in particular interested in object oriented Bayesian networks (Bangsø and Wuillemin, 2000). First, we define the common points and similarities between OOBNs and ontologies in order to set up a set of mapping rules allowing us to generate a prior OOBN by morphing ontology in hand and then, to use it as a starting point to the global learning OOBN algorithm. This latter will take advantages from both semantical data, derived from ontology which will ensure its good start-up and observational data which will ensure its progress. Then, we capitalize on the final structure resulting from the learning process to carry out the ontology enrichment.

This dissertation is organized in four chapters: In chapters 1 and 2, we will discuss our modeling requirements, respectively, probabilistic graphical models and ontologies and we will represent a survey on existing methods aiming to combine these two paradigms. In chapters 3 and 4, we present our research proposition and an overview of the contributions. Chapter 3 shows how the semantical richness of ontologies are used to generate a prior OOBN which will be used to launch the learning process. Chapter 4 shows how the result of the learning process might be a potential solution to enrich the ontology used initially. A detailed illustration that describes the whole process is given at the end of the fourth chapter.

Chapter 1

Probabilistic Graphical Models

1.1 Introduction

Probabilistic graphical models (PGMs) are a powerful tool for knowledge representation. They provide an effective framework for reasoning under uncertainty. *Bayesian networks* (BNs) are the result of the pioneering work of Judea Pearl in the 80's (Pearl, 1988) and one of the most used PGMs. Despite their great success, they are limited when dealing with large-scale systems. Thus, several extensions of these PGMs have been proposed in the literature in order to broaden their range of application.

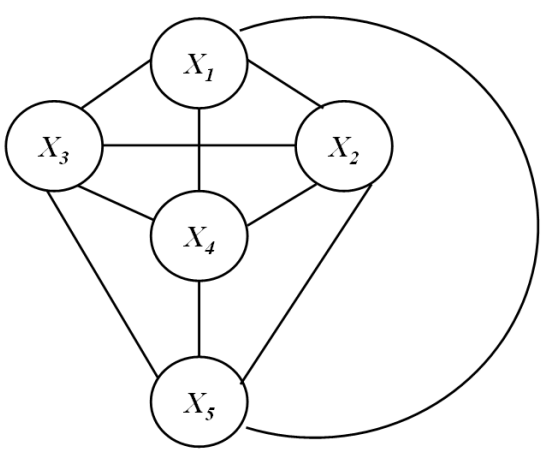
In this chapter, we provide some basic concepts in graph modeling and probability. Then, we will be interested in probabilistic graphical models. More precisely, we will focus on the family of Bayesian networks. We will define BNs and some of their extensions and we will look over their main cornerstones.

1.2 Basic concepts in graph modeling and probability

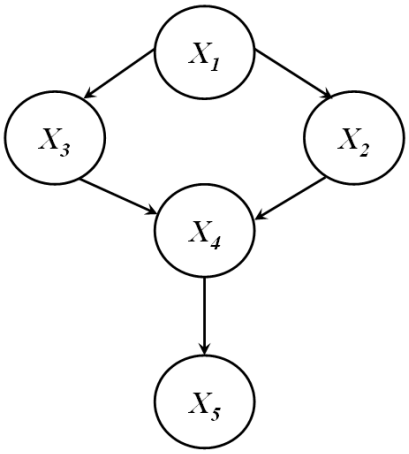
Probabilistic graphical models are founded on both graph and probability theories. Thus, it is useful to clarify some concepts and notions behind these theories before addressing these models.

1.2.1 Graphical models

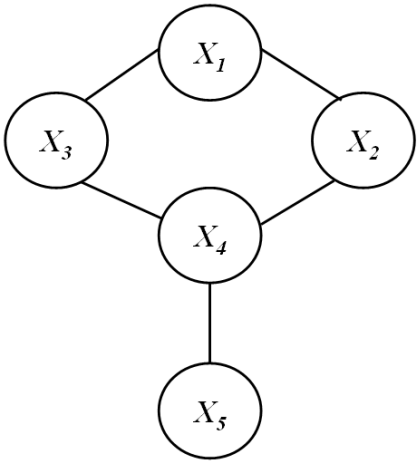
- A graph is defined by:
  - a set ϑ of vertices or nodes and,



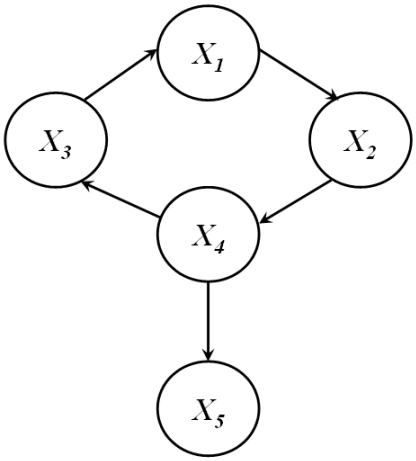
(a) A complete graph



(b) A directed acycle graph (DAG)



(c) An undirected graph (the skeleton of (a)



(d) A graph with cycle

Figure 1.1: Example of graphical models

– a set $E \subset \{(u, v) \| u, v \in \vartheta\}$ of edges or links.

- Two nodes that are connected by an edge are called adjacent.
- A graph in which every pair of nodes is connected by an edge is a complete graph *(e.g., figure 1.1(a))*.
- An edge may be undirected (unmarked link), directed (marked by a single arrowhead on the edge) or bidirected.

- A graph in which all edges are undirected is an undirected graph *(e.g., figure 1.1(c))*.
- A graph in which all edges are directed is a directed graph:
  - If we strip away all arrowheads from the edges of such a graph we obtain an undirected graph called its skeleton *(e.g., figure 1.1(c))*.
  - A directed path is a sequence of directed edges such that each edge starts with the vertex ending the preceding edge. The directed path is called a cycle if it begins and ends at the same variable *(e.g., figure 1.1(d): (X_1, X_2), (X_2, X_4), $(X_4, X_3$, $(X_3, X_1))$ is a cycle)*.
  - Directed graphs may include directed cycles (e.g., $X \to Y$, $Y \to X$) but not self-loops (e.g., $X \to X$).
  - A graph that contains only directed edges and no cycles is called directed acyclic graph (DAG)*(e.g., figure 1.1(b))*.
    * The set of the ancestors $An(X_i)$ of a node X_i is the set of nodes that can reach X_i by a directed path of length one or more *(e.g., figure 1.1(b): $An(X_5) = \{X_1, X_2, X_3, X_4\}$)*.
    * The set of the parents $Pa(X_i)$ of a node X_i, is the set of nodes that can reach X_i by a single arc pointing into X_i. A node with no parents is called a root *(e.g., figure 1.1(b): $Pa(X_5) = \{X_4\}$, X_1 is a root node)*.
    * The set of the children $Ch(X_i)$ of a node X_i is the set of nodes reachable by a single arc from X_i. A node with no children is called a sink *(e.g., figure 1.1(b): $Ch(X_4) = \{X_5\}$, X_5 is a sink)*.
    * The set of the descendants $De(X_i)$ of a node X_i is the set of nodes reachable by a directed path from X_i *(e.g., figure 1.1(b): $De(X_1) = \{X_2, X_3, X_4, X_5\}$)*.
    * The set of the non-descendants $Nd(X_i)$ of a node X_i is defined as $Nd(X_i) = (\vartheta \backslash (X_i \cup De(X_i) \cup Pa(X_i))$ *(e.g., figure 1.1(b): $Nd(X_2) = \{X_3\}$)*.

1.2.2 Probability and graphical models

Usually, graphical models are used in probabilistic modeling since they provide a convenient mean to represent probabilities assigned to states of the world in an uncertain context. Here we give some useful concepts of probability theory.

Let us denote random variables by an upper case letter (e.g., X, Y,...). Their tribes of events are denoted by corresponding calligraphic letters(e.g., $\mathcal{X}$, $\mathcal{Y}$,...).

Their assignments or states are denoted by corresponding lower-case letter (e.g., x, y,...).

Definition 1.2.1. ***(Joint probability distribution)***
The joint distribution of n random variables $X_1, \ldots, X_n$ is defined as follows:

$$\begin{aligned} P(X_1, \ldots, X_n): \quad & \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to [0,1] \\ & (x_1, \ldots, x_n) \longmapsto P(x_1, \ldots, x_n) \end{aligned}$$

In fact, the process of representing knowledge in Bayesian networks is related to the notions of conditional probability and conditional independence.

Definition 1.2.2. ***(Conditional probability)***
let $y \in \mathcal{Y}$ such that $P(Y = y) \neq 0$. The probability of $X = x$ (hypothesis) under the condition $Y = y$ (evidence) is:

$$P(x|y) = \frac{P(x,y)}{P(y)}$$

Using this formula and

$$P(y|x) = \frac{P(y,x)}{P(x)}$$

we can easily derive the ***Bayes' inversion formula*** *defined as follows:*

$$P(x|y) = \frac{P(y|x).P(x)}{p(y)}$$

Definition 1.2.3. ***(Independence)***
Two random variables X and Y are called independent (we note: $X \perp Y$) if the outcome of X has no effect on the outcome of Y and vice versa.
Therefore:

$$P(X|Y) = P(X), P(Y|X) = P(Y)$$

and the joint distribution of X and Y is:

$$P(X,Y) = P(X).P(Y)$$

Definition 1.2.4. ***(Conditional independence)***
Let three random variables X, Y and Z. We say that X and Y are conditionally independent to Z (we note: $X \perp Y|Z$) if $P(X,Y|Z) = P(X|Z).P(Y|Z)$

1.3 A survey on probabilistic graphical models

Probabilistic graphical models (Koller and Friedman, 2009) allow us to get a compact encoding of a complex distribution over a high-dimensional space using a graph-based representation. In this section ,we will be, in particular,interested in the family of Bayesian networks. We start by defining BN then we will look after some of their extensions.

1.3.1 Bayesian network definition

Formally, a Bayesian network (BN) $B = (G, \Theta)$ is defined by:

- A graphical (qualitative) component : a directed acyclic graph (DAG) $G = (V, E)$, where V is the set of vertices (nodes) represents n discrete random variables $X = \{X_1, \ldots, X_n\}$, and E is the set of directed edges (arcs) corresponds to conditional dependence relationships among these variables.

  Example 1.3.1. *Figure 1.2 is an example of a BN (Pearl, 2000). It illustrates a BN having five random variables and dependencies among them. X1 is the season of the year and it can take one of four values: spring, summer, fall, or winter. All other variables are binary: X2 describes whether the rain falls, X3 describes whether the sprinkler is on, X4 describes whether the pavement would get wet and X5 describes whether the pavement would be slippery.*

  This model encodes the following conditional independence assumption:

  Property 1.3.1. ***(Markov assumption)***

  Each variable X_i in G is independent of its non descendants given its parents.

  Example 1.3.2. *For our example (figure 1.2), knowing X_4 renders X_5 independent of $\{X_1, X_2, X_3\}$*

  Pearl introduced a graphical criterion for determining whether two variables are independent conditionally to a set of variables, namely the d-separation (Pearl, 1988).

  Before defining this criterion, let's present the three basic connection structures between variables:

  – Serial connection (chain) ($X \rightarrow Z \rightarrow Y$) and Diverging connection (fork)($X \leftarrow Z \rightarrow Y$): If we know the value of Z , then X has no influence on Y and vice versa. We say that X and Y are dependent, but they are conditionally independent given Z, and we note: $X \perp Y | Z$.

– Converging connection (collider, also called V-structure) ($X \rightarrow Z \leftarrow Y$): If we know the value of Z , then evidence on X influences Y and vice versa. We say that X and Y are independent, but they are conditionally dependent given Z, and we note: $X \perp Y$ (but$X \not\perp Y|Z$).

Thus, the d-separation criterion is defined as follow (Pearl, 1988):

Definition 1.3.1. ***(d-separation)***

1. *Let G = (V,E) be a DAG and let X, Y and Z be three distinct variables in V. Then, X and Y are said to be d-separated by Z , if for all paths between X and Y, there is an intermediate variable Z such that either:*
   (a) *the connection is serial or diverging and the state of Y is known, or*
   (b) *the connection is converging and neither the state of Y , nor any of its descendants, are known.*
2. *Let G = (V,E) be a DAG and let X, Y and Z be three disjoint subsets of V. Then, X and Y are said to be d-separated by Z, if for every variable $X_i \in X$ and $Y_i \in Y$,X_i andY_i are d-separated by Z.*

Example 1.3.3. *In figure 1.2, $X = \{X_2\}$ and $Y = \{X_3\}$ are d-separated by $Z = \{X_1\}$ as both paths connecting X_2 and X_3 ($X_2 \leftarrow X_1 \rightarrow X_3$ and $X_2 \rightarrow X_4 \leftarrow X_3$) are blocked by Z (The first path presents a diverging connection having its middle node X_1 in Z. While the second path is a converging connection having its middle node X_4 out Z).*

- A numerical (quantitative) component : presents a set of parameters $\Theta = \{\Theta_1, \ldots, \Theta_n\}$ where each $\Theta_i = P(X_i|Pa(X_i))$ denotes the conditional probability distribution of each node X_i given its parents $Pa(X_i)$.These conditional probability distributions are stored and organized in tables named Conditional Probability Tables (CPTs).

Thanks to the Markov assumption (see property1.1), Bayesian networks allow as to represent the joint probability distribution (JPD) of $X = \{X_1, \ldots, X_n\}$ as a decomposition of a global function into a product of local terms depending only on the considered node and its parents in the graph via the chain rule of BN

$$P(X_i) = \prod_{i=1}^{n} (P(X_i)|Pa(X_i))$$

We can have different graphs that define the same probability distribution, these graphs are then called equivalent.

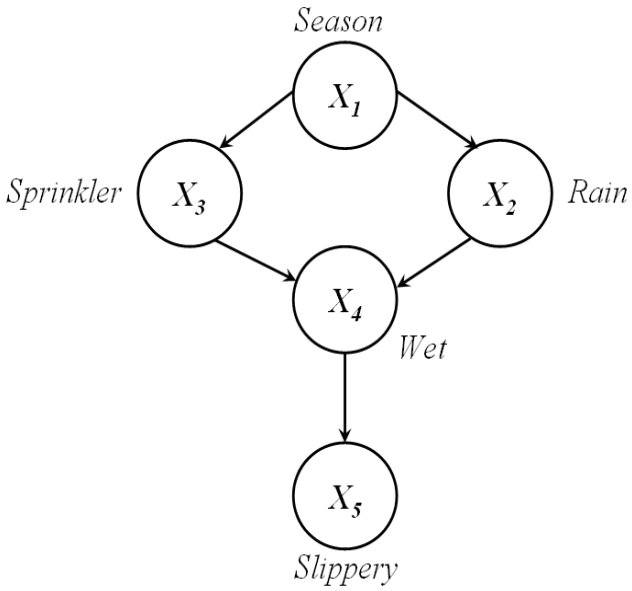


Figure 1.2: Example of a Bayesian network

Definition 1.3.2. ***(Markov equivalence)***
Two Bayesian networks B_1 and B_2 are equivalent if they define the same probability distribution.

Example 1.3.4. *Let's consider the figure 1.3. In this example we can easy demonstrate that G1 and G2 are equivalents by the decomposition of their joint probability distributions. We have:*

$$P(X_1, X_2, X_3)_{G1} = P(X_2|X_1) * P(X_1|X_3) * P(X_3)$$

and

$$P(X_1, X_2, X_3)_{G2} = P(X_2|X_1) * P(X_3|X_1) * P(X_1)$$

Thus:

$$\begin{aligned} PX_1, X_2, X_3)_{G2} &= P(X_2|X_1) * P(X_3|X_1) * P(X_1) \\ &= P(X_2|X_1) * \tfrac{P(X_1|X_3)*P(X_3)}{P(X1)} * P(X_1) \\ &= P(X_2|X_1) * P(X_1|X_3) * P(X_3) \\ &= P(X_1, X_2, X_3)_{G1} \end{aligned}$$

Similarly, we demonstrate that G3 is equivalent to G1 and G2. Yet these three networks are not equivalent to the v-structure G4. As $P(X_1, X_2, X_3)_{G4} = P(X_1|X_2, X_3)*$ $P(X_2) * P(X_3)$ and $P(X_1|X_2, X_3)$ can not be simplified.

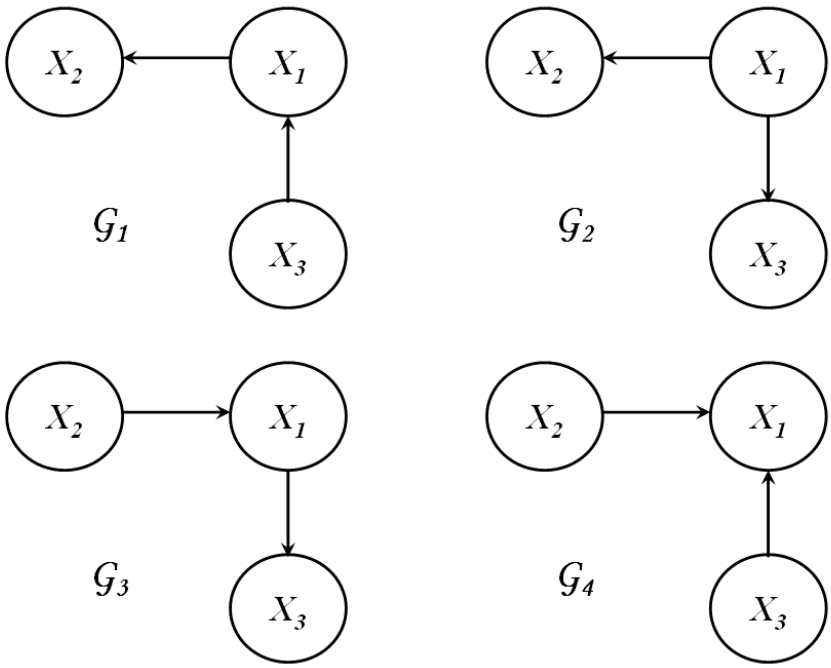


Figure 1.3: Markov equivalence

A graphical criterion for determining the equivalence of two DAGs was demonstrated by Verma and Pearl (Verma and Pearl, 1990) and expressed through the following theorem:

Theorem 1.3.1. *Two DAGs are equivalent if and only if they have the same skeletons and the same v-structures.*

A Markov equivalence class is defined as a set of equivalent Bayesian networks and represented via a Completed Partially Directed Graph (CPDAG).

Definition 1.3.3. *A CPDAG has the same skeleton as all the graphs in the equivalence class and all its reversible edges (edges that do not belong to a v-structure and their inversion does not generate a v-structure) are undirected.*

Chickering (Chickering, 2002) proposes a method that allows the conversion of a DAG to its CPDAG representing its Markov equivalence class.

1.3.2 Extensions of Bayesian networks

Despite their great success, BNs have some weaknesses. Thus, several extensions have been proposed in the literature in order to broaden their range of application.

We can, for instance, mention *dynamic Bayesian networks* (Murphy, 2002)that enable to model a system whose state evolves over time, *hierarchical Bayesian networks* (Gyftodimos and Flach, 2002) that gives additional knowledge about variables structure by extending the classical formalism of Bayesian networks with composite nodes allowing the aggregation of simpler types. Pearl (Pearl, 2000) focuses on the semantics of intervention and its relation to causality through the causal Bayesian networks (CBN) framework. Another range of extensions introduces higher level structure by using either the relational logic or the object paradigm to enrich the classical formalism of Bayesian networks. Based on the relational logic, probabilistic relational models (PRM) are a more refined representation of BNs which were first defined by (Pfeffer, 2000) to describe statistical models of structured data. Object oriented Bayesian networks (Bangsø and Wuillemin, 2000) (Koller and Pfeffer, 1997) (OOBNs) are an extension of BNs using the object paradigm which allows a more compact representation of knowledge thanks to the several aspects of object oriented modeling. In what follows, we successively detail the causal Bayesian network and the object oriented Bayesian network frameworks.

Causal Bayesian Networks

Bayesian networks are also named belief networks, probabilistic networks,...and in some cases, they are wrongfully called causal networks. In fact, the main idea of causal discovery is to uncover causal relationships between a set of units (people, cars, countries, genes, etc.). These cause to effect relationships are presented via graphical models named causal models that describe how data is generated in a specific system. In the particular case where different edges of a BN express direct causal relationships between nodes, we speak about causal Bayesian networks (Pearl, 2000) (CBN). Causal models allow not only probabilistic inference but also reliable causal inference. This latter presents the process of computing the effect of manipulating some variables on the probability distribution of some other variables, rather than computing the effect of observing some variables on the probability distribution of some other ones.

We define observation and manipulation as follow:

- **Observation.** We call observation of X_i when we passively observe the information $X_i = x_i$.
- **Manipulation.** We call manipulation of X_i and we denote it $do(X_i)$ an intervention of an external agent forcing the value of X_i to be equal to x_i.

Example 1.3.5. *To make the difference between these two notions, let us consider the example of figure 1.2. Suppose that we want to find the probability that it rained,*

given that we "see" the grass wet. By referring to the Bayes rule, we can respond automatically to this request as we know that:

$$P(rain|wet) = \frac{P(wet|rain)P(rain)}{P(wet)}$$

However, if we want to know the probability that it rained given that we "make" the grass wet, The probability syntax will be unable to express our query, since we know intuitively that making the grass wet does not change the chance of rain. Thus the answer should be $P(rain)$. A mathematical framework is set up to derive such an intuitive answer.

Definition 1.3.4. *A Causal Bayesian Network denoted by CBN is a BN that contains an edge $A \rightarrow B$ if and only if A is a direct cause of B.*

This new semantic interpretation of the arcs in a CBN allows the calculation of the effect of performing an intervention on the system, that is known as causal inference (Spirtes, 2010).

A CBN respects the Causal Markov assumption:

Property 1.3.2. ***(Causal Markov assumption)***
A node is independent of its distant causes given just its direct ones.

Object Oriented Bayesian Network

In few words, OOBNs are an extension of BNs using the object paradigm. They aim to divide a large network (modeling a large and complex domain) into smaller pieces which simplifies the modeling task and allows a more compact representation of knowledge (thanks to the several aspects of object oriented modeling, such as inheritance, instantiation, etc.). Furthermore, they handle repetitive and hierarchical structures. The first definitions of what an OOBN is can be found in (Bangsø and Wuillemin, 2000) (Koller and Pfeffer, 1997). While, we should mention that these definitions are quite different. In (Bangsø and Wuillemin, 2000), the authors confront the two depictions and clarify the similarities and dissimilarities between them. In this part, we adopt the OOBN framework of Bangsø and Wuillemin (Bangsø and Wuillemin, 2000), (Bangsø et al., 2000). An Object Oriented Bayesian Network (OOBN) models the domain using fragments of a Bayesian network known as classes. Each class can be instantiated several times within the specification of another class.

Formally, a class T is a DAG over three, pairwise disjoint sets of nodes $(\mathcal{I}_T, \mathcal{H}_T, \mathcal{O}_T)$, such that for each instantiation t of T:

- $\mathcal{I}_T$ is the set of input nodes. All input nodes are references to nodes defined in other classes (called referenced nodes). Each input node have at most one referenced node, it has no parents in t and no children outside t.

- $\mathcal{H}_T$ is the set of internal nodes including instantiations of classes which do not contain instantiations of $\mathcal{T}$. They are protected nodes that can't have parents or children outside t.
- $\mathcal{O}_T$ is the set of output nodes. They are nodes from the class usable outside the instantiations of the class and they can not have parents outside t. An output node of an instantiation can be a reference node if it is used as an output node of the class containing it.

Internal nodes and output nodes (except those that are reference nodes) are considered as real nodes and they represent variables. In an OOBN, nodes are linked using either directed links (i.e., links as in standard BNs) or reference links. The former are used to link reference or real nodes to real nodes, the latter are used to link reference or real nodes to reference nodes. Each node in the OOBN has its potential, i.e. a probability distribution over its states given its parents. To express the fact that two nodes (or instantiations) are linked in some manner we can use construction links ($---$) which only represent a help to the specification.

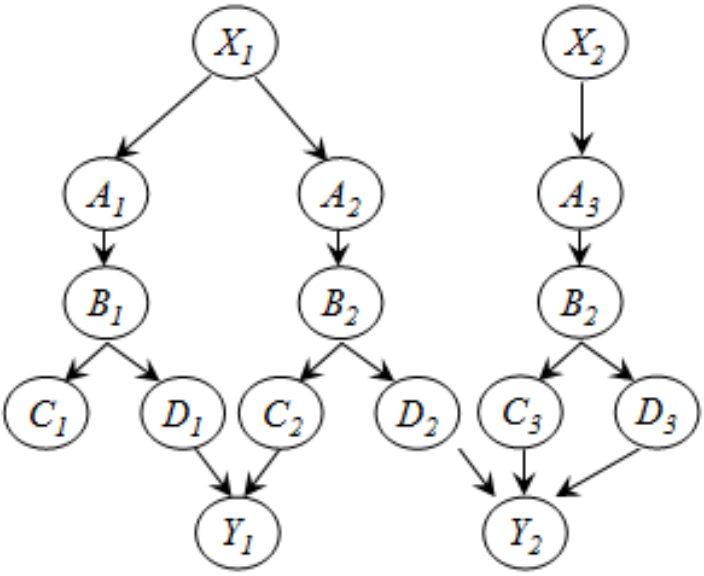


Figure 1.4: A Bayesian network with repetitive structures

Example 1.3.6. *In figure 1.4, we assume that X_1 and X_2 have the same state space and the conditional probability tables (CPTs) associated with all nodes la-*

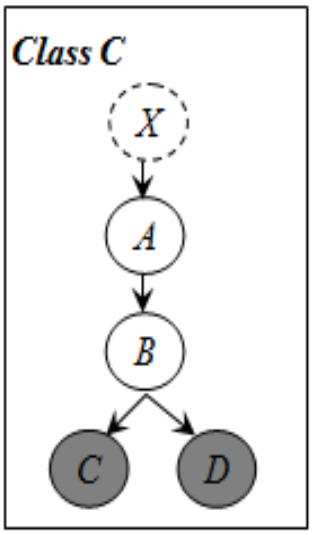


Figure 1.5: A class model for the repetitive structure in figure 1.4

beled A_i as well as nodes labeled B_i, C_i and D_i, where $i = \{1, 2, 2\}$, are identical. Hence, we have three copies of a same structure. When modeling an OOBN such a repetitive structure will be presented by a class (see figure 1.5), where the dashed node X is the input node (an artificial node having the same state space as X_1 and X_2), the shaded nodes C and D are output nodes (nodes accessible outside the object that may be parents of nodes outside), and A and B are the encapsulated nodes, they are invisible to the rest of the model.

Having this description, we can represent the BN of figure 1.4 using an OOBN model (see figure 1.6). The class C is instanciated three times and the nodes X_1, X_2, Y_1 and Y_2 are connected to the appropriate objects labeled $I.1, I.2$ and $I.3$.

In the extreme case where the OOBN consists of a class having neither instantiations of other classes nor input and output nodes we collapse to standard BNs.

Another powerful property of OO modeling is also considered when modeling an OOBN, namely, the inheritance between classes. Formally, a class S over $(\mathcal{I}_S, \mathcal{O}_S, \mathcal{H}_S)$ is a subclass of a class T over $(\mathcal{I}_T, \mathcal{O}_T, \mathcal{H}_T)$, if $\mathcal{I}_T \subseteq \mathcal{I}_S, \mathcal{O}_T \subseteq \mathcal{O}_S$ and $\mathcal{H}_T \subseteq \mathcal{H}_S$. The subclass inherits also the conditional probability tables (CPTs) of the corresponding node in its superclass unless the parent sets differ, or the modeler explicitly overwrites some CPTs.

Finally, we should mention that in some cases, some Input nodes do not reference any node. This may happen when no reference link is specified or when an

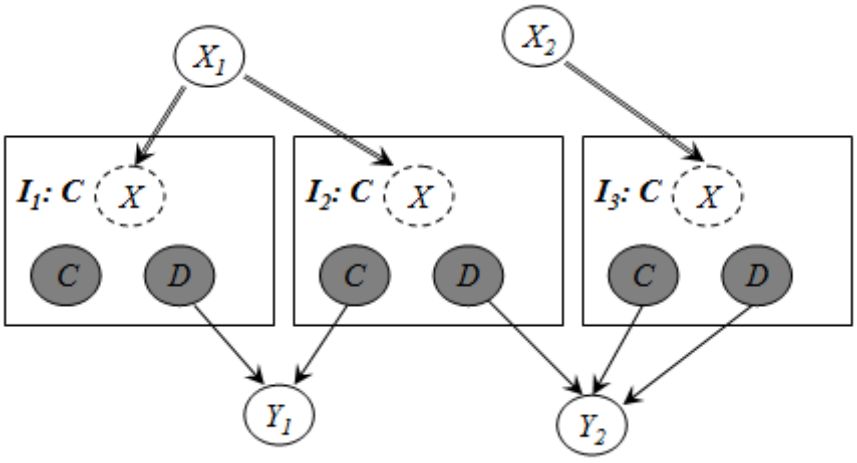


Figure 1.6: An OOBN representing the model shown in figure 1.4

instantiation of a subclass is used instead of an instantiation of its superclass (so we can have extra input nodes). That's why a default potential (i.e., a probability distribution over the states of an input node) is associated to each input node and is used in such cases.

If a class is instantiated more than once, the Object Oriented assumption must be satisfied.

Property 1.3.3. ***(Object Oriented assumption)***

All instances of a class are assumed to be identical w.r.t. both parameters and structure.

Object Oriented Bayesian Networks (OOBNs) are a convenient representation of knowledge containing repetitive structures. So they are a suitable tool to represent dynamic Bayesian networks as well as some special relations which are not obvious to represent using standard BNs (e.g., examine a hereditary character of a person given those of his parents) and this is by constructing a generic class that can be instanciated as many time as needed. Furthermore, they allow to represent hierarchy between classes thanks to the inheritance mechanism.

Torti et al. (Torti et al., 2010) merge the OOBN and the PRM framework in order to preserve the advantages of both. In fact, PRMs lack fundamental mechanisms related to object paradigm. In return, they represent others advantages which are not allowed by the formalism of OOBNs (e.g. multiple references).

1.4 Main issues of probabilistic graphical models

All PGMs share two fundamental cornerstones: learning of both parameters and structure and inference. We will discuss these points for the models defined above.

1.4.1 Structure learning

Learning the structure of a PGM consists on providing a network structure which fits the best possible way to the observed data.

Bayesian networks structure learning

Learning the Bayesian network structure is an unsupervised learning problem which is known to be a NP-Hard problem (Chickering et al., 1994). The literature (Chickering, 2002; Cooper and Herskovits, 1992; Henrion, 1988; Spirtes et al., 2001; Buntine, 1996) involves two main families of structure learning algorithms, namely score-based and constraint-based. For the first, the idea is to evaluate how well the structure fits to the data using a score function. So, those algorithms search for the structure that maximizes this function. While, constraint-based algorithms look for independencies (dependencies) in the data, using statistical tests then, try to find the most suitable graphical structure with this information.

Besides the **Markov assumption**, other assumptions are required before detailing these approaches.

1. **Causal sufficiency assumption:** The set of variables X is sufficient to represent all the conditional dependence relations that could be extracted from data. That's mean that we assume the absence of latent variables.

2. **Faithfulness assumption:** We say that a joint probability distribution P and a DAG G are faithful to each other if and only if every independence present in P is entailed by G and the Markov assumption.

Score-based learning. The idea is to find the network that fits best to data while being as simple as possible. Thus, score-based algorithms assign a score to each possible network based on these two criteria, to search after through the space of graphs for the network that maximizes this score. Several scoring functions have been developed to assess the goodness-of-fit of a particular model, such as the Bayesian information criterion (BIC) (Schwarz, 1978) and the Bayesian score (BS) (Cooper and Herskovits, 1992; Heckerman, 1998). These scoring functions share two interesting properties:

- **Score-equivalent scoring function:** A scoring function is score-equivalent if it assigns the same score to equivalent structures (see definition 1.6).

- **Decomposable scoring function:** A scoring function is decomposable if it can be written as a sum of measures, each of which is a function only of one node and its parents.

Several algorithms have been proposed such as Maximum Weight Spanning Tree(MWST), K2, Greedy Search (GS), Greedy Equivalence Search (GES), etc. Most of these algorithms was extended for the case of incomplete data such as MWST-EM (Leray and Francois, 2005), SEM (Friedman, 1998), Ges-EM (Borchani et al., 2006), etc. A more detailed presentation of these algorithms is given in (Naïm et al., 2004).

Constraint-based learning. Several algorithms (Naïm et al., 2004) have been proposed either by Pearl and Verma (IC and IC* algorithms) or by Spirtes, Glymour and Scheines (SGS, PC, IC,...). All those algorithms share the same structure:

i. Construct a complete undirected network from the given data.

ii. Remove edges and detect V-structures using conditional independence tests.

iii. Try to orient more edges using the already oriented ones.

Causal Bayesian networks structure learning

When learning the structure of CBNs, we focus on finding causal relationships among variables rather than finding probabilistic dependencies. Both families of algorithms used to learn BNs are also used to learn CBNs. However, we will present this algorithms depending on the data used to learning: experimental and (or) observational.

Using only observational data, seems not to be a good way for learning CBNs: The process will be similar to this of learning BNs: learning the CPDAG and then choosing a possible complete instantiation in the space of its Markov equivalence class. In this case, causal relationships will be partially found since there is only one true CBN that represents this mechanism (see figure 1.7). Consequently, approaches that focus on learning CBN aim to directed the remaining edges in order to find the correct causal influence. Thus, two approaches were developed: The first uses only experimental data, while the second uses both experimental and observational data.

Tong and Koller (Tong and Koller, 2002) and Murphy (Murphy, 2001) proposed active learning approaches that learn the CBN structure from perfect experimental data gathered by structural experiments. While Cooper and Yoo (Cooper and

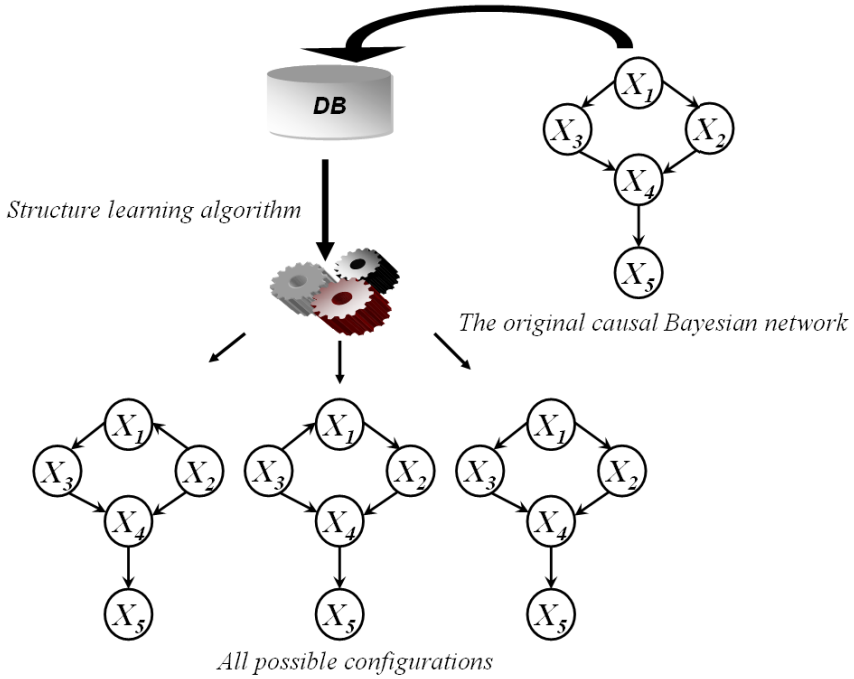


Figure 1.7: A CBN generates a database, that, in its turn, can be used to learn several BNs structures of which only one is the right CBN.

Yoo, 1999) proposed a method that allows to learn causal structure from mixture of imperfect observational and experimental data gathered also by structural experiments.

Meganck et al. (Meganck et al., 2006a)proposed a greedy approach to learn CBNs from perfect observational and experimental data. First, it learns the CPDAG from the perfect observational dataset using traditional structure learning techniques. Then, the second phase uses the CPDAG resulting from the first phase and elements from decision theory in order to direct the remaining edges and get the correct CBN modeling the original data. They proposed also the Uncado (Unsure Causal DiscOvery) algorithm (Meganck et al., 2008) to learn CBN from incomplete observational data and interventions, and MyCaDo++ algorithm (Meganck et al., 2006b) to learn the complete structure of a causal latent network.

Another research that aims to learn causal Bayesian networks from incomplete observational data and interventions was proposed by Borchani et al. (Borchani et al., 2007) as extension of the GES-EM algorithm proposed in (Borchani et al., 2006). They developed two approaches: an adaptive one, where interventions are done sequentially, and a non-adaptive one, where interventions are executed simultaneously.

Algorithm 1: OO-SEM[1](Langseth et al. (Langseth and Nielsen, 2003))

Input: A data base $\mathcal{D}$
Output: An OOBN B

begin

a) Let B_S^0 be the prior OOBN model.

b) **for** $n = 0, 1, \ldots$ *until convergence* **do**

1. Compute the posterior $P(\Theta_{B_S^n}|B_S^n, o)$.

2. $B_S^{n,0} \leftarrow B_S^n$.

3. **for** $i = 0, 1, \ldots$ **do**

i) Let B_S be the model which is obtained from $B_S^{n,i}$ by employing either none or exactly one of the operations add-edge and remove-edge, for each instantiation I; each edge involved must have a node in both I and in the encapsulating context of I (directed into I). The *OO* assumption is disregarded.

ii) For each node X, which is a child of an input node Y' (found in step (i)) in instantiation I_j, determine if $I_k.X$ has an input node as parent with the same state space as Y', for all $k \neq j$ where $T(I_k) = T(I_j)$. If this is the case, use the BDe score to determine if they should be assigned the same CPT (due to the OO assumption); otherwise introduce default potentials to ensure that they have the same CPTs. Let B_S' be the resulting network.

iii) For each class C_l in B_S' employ the operations add-edge or remove-edge w.r.t. the nodes in the class (excluding the input set) based on the candidate interface found in step (ii). Note that edges from instantiations encapsulated in C_l into nodes defined in C_l are also considered in this step. Let B_S'' be the resulting OOBN.

iv) $B_S^{n,i+1} \leftarrow B_S''$.

4. Choose $B_S^{n+1} \leftarrow B_S^{n,i}$ that maximizes $Q(B_S^{n,i} : B_S^n)$

5.**if** $Q(B_S^n : B_S^n) = Q(B_S^{n+1} : B_S^n)$ **then**

Return B_S^n.

[1]If the dataset is complete, then the outer loop will be evaluate once.

Object Oriented Bayesian networks structure learning

Few works have been proposed in the literature in order to learn the structure (Bangsø et al., 2001) (Langseth and Nielsen, 2003) of an OOBN. The standard approach proposed by (Langseth and Nielsen, 2003) is the OO-SEM algorithm (see algorithm 1). This algorithm is based on a prior expert knowledge which allows to get a partial specification of the OOBN by grouping nodes into instantiations and instantiations into classes. Then, on the basis of this prior, the learning process adapts the SEM algorithm (Friedman, 1998) in order to learn from a complete or incomplete dataset the OOBN structure that fits best to the data. The algorithm starts by learning the interfaces of the instantiations. Then, the structure inside each class is learned based on the candidate interfaces founded in the previous step.

Structure learning is performed with respect to the Object Oriented assumption. Suppose that our OOBN contains two instances $I1$ and $I2$ of the class C. Due to the OO assumption, if the node A of instance $I1$ is assigned an input node X as parent during model search, and that X references the node Y, then the search algorithm should find a node that has the same influence on $I2.A$ as Y has on $I1.A$. The search for this node must cover all output nodes in all instances in the encapsulating class as well as all its simple nodes. Thus, the complexity of finding the best candidate for all instances is exponential in the number of instances. In addition, this node may be non-existent and in this case the default potential for the input node is used.

Learning in object oriented domains allows to reduce the search space and the number of local maxima encountered however, it remains an NP-hard problem. In fact, as indicated in the previous paragraph, the main computational phase in the OO-SEM algorithm consists in finding the interfaces of instantiations, which is exponential in the number of instantiations. So, this information may be also elicited from domain experts, if the experts are able to specify exactly all input and output nodes for each instance, then, we have only to identify the structure inside classes.

1.4.2 Parameters learning

When learning the parameters of a PGM, we assume that the structure is known and we try to find the set of probability values that best represents the data.

BNs and CBNs Parameters Learning

The task of learning the parameters of a BN (CBN too) is to compute the CPTs for a fixed DAG. Many approaches have been proposed for learning CPTs from

complete or incomplete data (see (Heckerman, 1998) for more details). Some times, when there is no available data (or few), expert knowledges are required to estimate conditional probabilities, this is known as elicitation of probabilities. (Renooij, 2001) is an example of such works.

OOBNs Parameters Learning

The proposed method (Langset and Bangsø, 2001) learns in the class specification instead of in each instantiation. With respect to the OO assumption, all instantiations of a class are identical. Thus, the CPTs are only represented in the class which significantly reduces the number of parameters to learn. However, for the particular case of class hierarchy, CPTs inherited from the superclass may be redefined in its subclasses if ever the parent sets are different. In (Langset and Bangsø, 2001), the authors show how to learn the parameters of an OOBN in both cases of missing and no missing data.

1.4.3 Reasoning with PGMs

For PGMs, reasoning stands for probabilistic inference. It consists in providing algorithms which are able, having a fixed structure and parameters, to compute the posterior probability of some variables given evidence on others.

Reasoning with BNs

A Panoply of algorithms have been proposed for doing probabilistic inference (propagation) with BNs. such a process looks for the impact of a certain information regarding some variables, known as an evidence, on the remaining ones. The first algorithms are based on message-passing architecture and were proposed by Pearl (Pearl, 1982), and Kim and Pearl (Kim and Pearl, 1983). An extension of these algorithms, that are limited to trees, is found in several works like the Lauritzen and Spiegelhalter's method of join-tree propagation and the method of cut-set conditioning (Pearl, 2000) that provide propagation in general networks. All these algorithms are exact, however, inference in general networks is known to be NP-hard (Cooper, 1990), which means that in some cases (a huge number of nodes, a well connected DAG,...) the computational complexity of these algorithms may exceed reasonable bounds thus, a set of approximate algorithms was proposed to be used in such case.

Reasoning with CBNs

CBNs, from their side, allow not only probabilistic but also causal inference. They give the possibility to estimate the influence of external intervention on any variable

of the graph. Pearl (Pearl, 2000) makes the distinction between conditioning and intervening by introducing the do-operator: The intervention (also known as manipulation, experiment, or action) $do(X = x)$ sets a subset X of variables to constants x. In fact, thanks to Bayes' theorem the principle of conditional probability is symmetric. So, we can not use this principle to represent causal relations that are asymmetric, The do-operator is then presented to overcome this problem: If X causes Y, then we have:

$$P(Y = y|do(X = x)) = P(Y = y|X = x)$$

This means that manipulating the value of X affects the value of Y.

and

$$P(X = x|do(Y = y)) = P(X = x)$$

This mean that manipulating Y will not affect X.

Fixing a set of variables M to some constants $do(M = m)$ modifies the domain that is modeled by a CBN as well as the JPD that is used to model it.

Algorithm 2: Construction of the underlying BN: BN_I (Bangsø et al. (Bangsø and Wuillemin, 2000))

Input: An OOBN
Output: A BN

begin

1. Let BN_I be the underlying BN of The class T, where T is the global OOBN class (the environment for the classes to be instantiated in).

2. Add a node to BN_I for each input node, output node and normal node in I.

3. Add a node to BN_I for each input node, output node and normal node of the instantiation to the node name (Instantiation-name.node-name). Do the same for instantiations contained in these instantiations and so on.

4. Add a link for each normal link (i.e. directed link) in I, and repeat this for all instantiations as above.

5. For each reference tree, merge all the nodes into one node. This node is given all the parents and children (according to the normal links) of the nodes in the reference tree as its family. Note that only the root of the tree can have parents, as all other nodes are references to this node.

Reasoning with OOBNs

To deal with probabilistic inference, an OOBN can be translated into a multiply-sectioned Bayesian network (Xiang et al., 1993)(for more details see (Bangsø and Wuillemin, 2000)) or compiled in order to construct the underlying BN. This latter is constructed following algorithm 2.

1.5 Conclusion

Thanks to the intuitive formulation of dependence relationships between variables and the inference capabilities of Bayesian networks, their use as knowledge representation tools increases more and more in the machine learning community, and several extensions have been proposed in order to broaden their range of application.

In this chapter, we introduced basic definitions and concepts related to BNs and we showed some of their extensions. In particular, we focused on CBNs and OOBNs. Then, we presented an overview of prevalent issues related to these PGMs and we described the major existing approaches for learning them from data.

In the next chapter, we introduce another paradigm used especially in the knowledge engineering domain known as ontology.

Chapter 2

Ontologies

2.1 Introduction

Ontology is a philosophical term that deals with the nature of being, it first appeared in the Aristotle's works on metaphysics. In the early 1990's, this term was integrated in the artificial intelligence field, especially in the knowledge engineering domain in order to specify the content of shared knowledge. Since then, considerable progress has been made in developing the conceptual basis for building ontologies. In this chapter, we will first define ontology. Then, we will make an overview on ontology evolution methods. Finally, we will outline some research directions aiming to combine ontologies and PGMs.

2.2 Basics on ontology

This section is devoted to defining ontologies and describing some ontology languages.

2.2.1 Ontology definition

An ontology defines a set of concepts and relationships among them. It is used to model and share a domain of knowledge or discourse. Many definitions of what an ontology is have been proposed, and the most cited one is given by Gruber (Gruber, 1993):

"An ontology is an explicit specification of a conceptualization."

That is, an ontology is a description of a set of representational primitives with which to model an abstract model of a knowledge domain. The representational

primitives are the components of the ontology whose definition depends on the knowledge modeling technique used.

Definition 2.2.1. *Formally, we define an ontology* $\mathcal{O} = \langle \mathcal{C}p, \mathcal{R}, \mathcal{I}, \mathcal{A} \rangle$ *as follows:*

- $\mathcal{C}p = \{cp_1, \ldots cp_n\}$ *is the set of n concepts (classes) such that each* cp_i *has a set of k properties (attributes)* $\mathcal{P}_i = \{p_1, \ldots p_k\}$.
- $\mathcal{R}$ *is the set of binary relations between elements of* $\mathcal{C}p$ *which consists of two subsets:*
  - $\mathcal{H}_\mathcal{R}$ *which describes the inheritance relations among concepts.*
  - $\mathcal{S}_\mathcal{R}$ *which describes semantic relations between concepts. That is, each relation* $cp_i s_R cp_j \in \mathcal{S}_\mathcal{R}$ *has* cp_i *as a domain and* cp_j *as a range.*
- $\mathcal{I}$ *is the set of instances, representing the knowledge base.*
- $\mathcal{A}$ *is the set of the axioms of the ontology.* $\mathcal{A}$ *consists of constraints on the domain of the ontology that involve* $\mathcal{C}p$, $\mathcal{R}$ *and* $\mathcal{I}$.

Example 2.2.1. *Suppose that we want to create an ontology of persons, a man is a person. So Man is a concept of this ontology. Woman is also a concept of this ontology. A person is characterized by his name and age thus,* $\mathcal{P}_{Person} = \{name, age\}$.

Concepts may be organized into a superclass-subclass hierarchy: $cp_2 \subseteq cp_1$. We say that subclasses are subsumed by their superclasses. This is presented by the "is_a" relation: *Man is_ a subclass of Person. The cpncept Man inherits all the properties of the concept Person thus,* $\mathcal{P}_{Man} = \mathcal{P}_{Person} \cup \{Militaryservice\}$ *This says that all members of the class Man are members of the class Person. (see figure 2.1).* Just like concepts, We may use a mechanism of hierarchy between properties.

Instances represent objects in the domain in which we are interested. They instantiate concepts and relations: *Mary who is a woman, having 28 years old and Lam as Maiden name "Is_ Married_ To" Peter how is a man, having 30 years old and who has done its Military service. James who is a man, having 50 years old and who has done its Military service "Is_ Father" of Paul how is a man, having 15 years old and who has not done its Military service. (see figure 2.3)*

As we said above, axioms consist in constraints on the domain of the ontology, e.g., constraints on properties such as domain restriction, constraints on relations such as using hierarchy and defining a set of characteristics of relations.

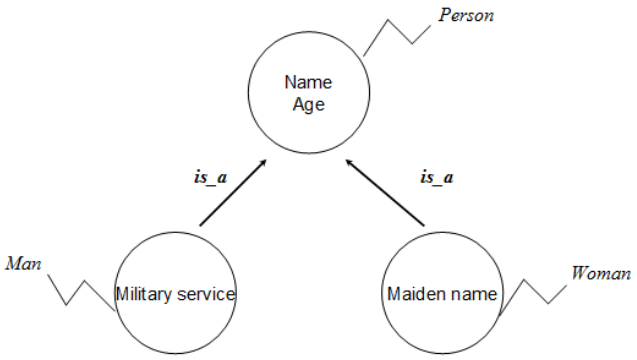


Figure 2.1: Representation of concepts

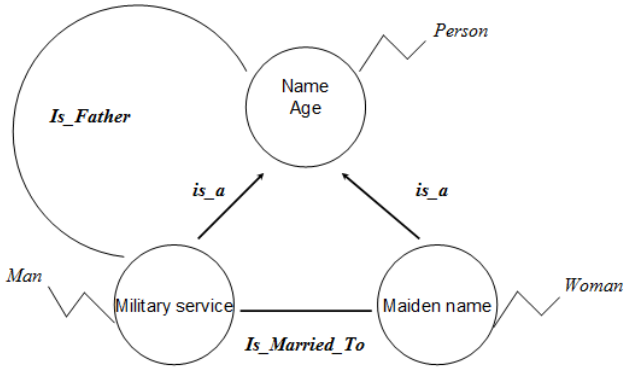


Figure 2.2: Representation of properties

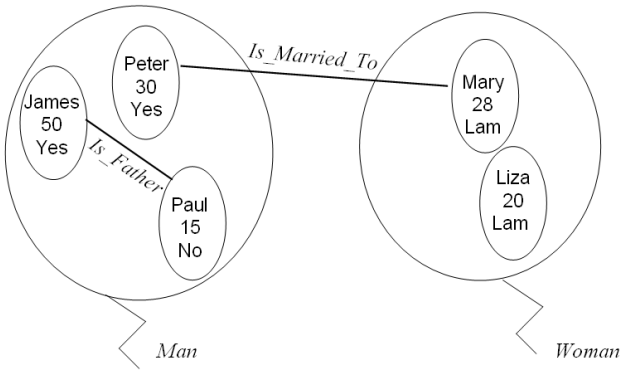


Figure 2.3: Representation of instances

For example, the property "Is_ Married_ To" might link the concept Man to the concept Woman, and it is a symmetric relation. The property "Is_ Father" might

link the concept Man to the concept Person, but it is asymmetric.(see figure 2.2)

In (Gruber, 1993), Gruber used frames and first order logic to model ontologies. Another AI-based modeling approach uses the description logic which is a logical formalism divided into two parts (Gómez-Pérez et al., 2004): the Terminological Box (TBox) and the Assertional Box (ABox). The former contains the definitions of concepts and roles while the latter contains the definitions of individuals (instances). Based on this formalism, ontologies may be represented using three kinds of components: concepts to represent classes of objects, roles to either describe binary relations between concepts or to describe properties of concepts and individuals that are instances of classes.

A relevant question to ask is: What an ontology is for? An ontology aims to gather general knowledge related to a specific domain. Then it shares this knowledge between humans, systems, as well as software agents while allowing its reusability.

To guide the development of ontologies, Gruber proposed in (Gruber, 1995) a set of design criteria for ontologies.

- **Clarity:** An ontology should convey the intended meaning of defined terms unambiguously.
- **Coherence:** An ontology should be coherent and allows for meaningful inferences that are consistent with definitions and axioms.
- **Extendibility:** When designing the ontology we should take into account future possible extensions without a need for revising definitions
- **Minimal encoding bias:** The conceptualization should not depend on a particular symbol-level encoding.
- **Minimal ontological commitment:** An ontology should make as few as possible of assumptions about the domain being modeled.

Usually ontologies are equated with taxonomic hierarchies of classes. In fact, ontologies are not limited to these forms, they have been used to describe artifacts with different degrees of structure and to model the domain of knowledge in a deeper way. Axioms present a basic component that do constrain the possible interpretations for the defined terms (Gruber, 1995). Thus the ontology community distinguishes lightweight from heavyweight ontologies. Lightweight ontologies are closer to taxonomies and consist of concepts, concept taxonomies, relationships (not only restricted to the is-a relation) between concepts and properties defined on concepts, whereas heavyweight ontologies add axioms and constraints to lightweight ontologies to provide more semantic modeling capabilities (Gómez-Pérez et al., 2004).

To encode ontologies, we are in need of formal languages (Gómez-Pérez et al., 2004). Some of them are based on a combination of frames and first order logic such as CycL, OCML, FLogic, etc. Some other languages were developed to use the characteristics of the Web and they are known as ontology markup languages as their syntax uses a markup scheme to encode knowledge, most commonly XML. OWL, the formal W3C recommendation for representing ontologies, is an example of these languages.

2.2.2 Ontology languages

Ontologies can be very useful for a community as a way of structuring and defining the meaning of the metadata terms that are currently being collected and standardized. They figure prominently in the emerging Semantic Web as a way of representing the semantics of documents and enabling the semantics to be used by web applications.

In fact, the Semantic Web extends the Web architecture with semantic tools in order to give an explicit meaning to the information and make it easier for machines to automatically process and integrate information available on the Web.

The Semantic Web is based on the Web technologies and concepts such as XML and XML Schema to which new technologies, specific to the Semantic Web are added. In fact, XML imposes no semantic constraints on the meaning of the documents, it only provides a surface syntax for structured documents. XML Schema extends XML with datatypes and restricts the structure of XML documents by defining their structure and content. However, all Web technologies do not support semantic purposes, this latter are then invoked by the Semantic Web technologies predominately, RDF and RDF Schema.

The W3C established the Resource Description Framework (RDF)[1] recommendation to describe Web resources through RDF triples of subject, predicate (also called property) and object. each RDF triple can be encoded in XML and represented by an RDF graph where the subject and object are nodes linked by a directed arc that corresponds to the predicate as shown in figure 2.4. However, the RDF data model does not provide modeling primitives for defining the relationships between properties and resources. Thus, the RDF Schema (RDFS)[2] has emerged. RDF Schema represents the vocabulary for describing RDF resources and it is considered as a simple ontology language. The latter is not able to achieve interoperation between numerous, autonomously developed and managed schemas as it is not expressive enough to catch all the relationships between classes and properties. A web ontology language which can formally describe the semantics

[1] http://www.w3.org/TR/rdf-concepts/
[2] http://www.w3.org/2001/sw/wiki/RDFS

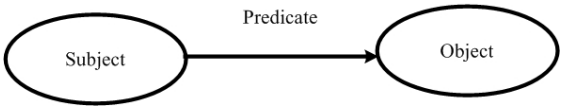


Figure 2.4: The RDF graph

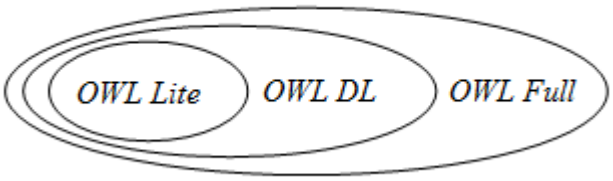


Figure 2.5: The OWL layers

of classes and properties used in web documents is required. This latter must go beyond the basic semantics of RDF Schema in order to allow machines to perform useful reasoning tasks on the Web resources.

Several languages have been designed to meet this need for a Web Ontology Language, such as Ontology Inference Layer (OIL)[3] and DAML+OIL[4] that use description logics to give clear semantics to modeling primitives. Then, the W3C Web Ontology Working Group created the OWL language. This language is derived from the DAML+OIL language, and it is built upon RDFS. OWL offers a richer vocabulary than RDF schema, among others:

- The description of relationships between classes (e.g. disjointness)
- The cardinality accuracy (e.g. "exactly one")
- The characteristics of properties (e.g. symmetry)

The underlying formalism for reasoning about data in the OWL ontologies is description logics. The first version was published in 2004 and the current version of OWL, also referred to as OWL 2, was published in 2009[5]. OWL is part of the growing stack of W3C recommendations related to the Semantic Web.

OWL has three named sublanguages (McGuinness and Harmelen, 2004): OWL Lite, OWL DL, and OWL Full, knowing that OWL Lite is a syntactic subset of OWL DL and OWL DL is a syntactic subset of OWL full (see figure 2.5).

[3]http://dret.net/glossary/oil
[4]http://www.w3.org/TR/daml+oil-reference
[5]http://www.w3.org/2001/sw/wiki/OWL

1. **OWL Lite:** with a restricted expressivity, it captures the most essential and basic features of ontologies as it is used to build classification hierarchies (taxonomies) by means of some simple constraints.

2. **OWL DL:** more complex than OWL lite but also more expressive. OWL DL is so named due to its correspondence with description logics that forms the formal foundation of OWL. It includes all OWL language constructs used under certain restriction in order to guarantee computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time).

3. **OWL Full:** the most complex and expressive version but with no computational guarantees, it is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

2.3 Main issues of ontologies

Ontologies represent not only a body of structured knowledge but also the basis for deductive reasoning. in this section, we start by defining the ontologies building process then, we explore their reasoning abilities.

2.3.1 Ontology building process

During the last few years, increasing attention has been focused on ontologies and ontological engineering. By ontological engineering we refer to the set of activities that concern the ontology life cycle (see figure 2.6) covering its design, deployment, up to its maintenance which is becoming more and more crucial to ensure the continual update of the ontology toward possible changes.

Initially, ontology building (covering the construction and maintenance stages) has been considered as a manual process, made by domain experts and lacks of theorical foundation. However, due to the large and complex structures of ontologies, learning based human intervention becomes a hard and error-prone process. Therefore, it needs to be (semi) automated to improve both its speed and reliability. To carry out this automatization, several methods have been proposed which are based on various techniques. Natural language processing (NLP) and text mining techniques are the most widely used. Machine learning techniques are also used, commonly, together with linguistic techniques (Ruiz-Casado et al., 2005; Cimiano and Staab, 2005).

Several ontology change management activities (Flouris et al., 2006a) were presented to maintain ontologies and update them given possible changes. Among these activities, we find:

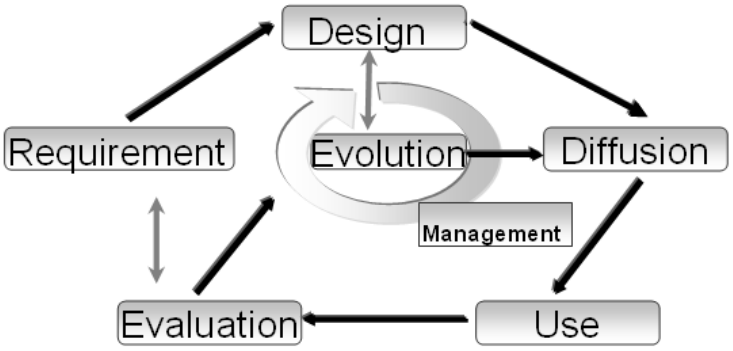


Figure 2.6: The ontology life cycle

- **Ontology evolution process:** is the process of modifying an ontology in response to a certain need of change.
- **Ontology versioning process:** is the process of modifying an ontology while keeping accessible and intact all previous versions.
- **Ontology merging process:** aims to construct a new coherent ontology from other ontologies covering the same topic and having highly overlapping or even identical domains.
- **Ontology integration process:** aims to construct a new coherent ontology from other ontologies covering related domains. The domain of discourse of the new ontology is then more general than the domains of source ontologies.
- **Ontology translation process:** it refers to the process of changing the formal representation of an ontology from one language to another.

A more detailed explication of these change activities and other ones is found in (Flouris et al., 2006a). We will focus on the task of ontology evolution which is considered as an interesting area of research.

Definition 2.3.1. *The Ontology evolution process (Stojanovic et al., 2002) is defined as the timely adaptation of an ontology in response to a certain change in the domain or its conceptualization.*

With reference to the definition of Gruber (see subsection 2.2.1), changes may arise in the domain, in the conceptualization or in the specification.

Changes in the domain are the consequence of the dynamism of the real world. Changes in the conceptualization are motivated by several events such as a new

information that may enrich the ontology, a change in the use of the ontology, etc. Changes in the specification refer to changes in the way the conceptualization is formally recorded such as changes in the representation language. Thus, ontology translation deals with this type of changes (Flouris et al., 2006a).

More precisely, evolution can be of two types (Khattak et al., 2009):

- **Ontology population process** consists in introducing new instances of the ontology concepts and relations.

  Instantiation is important in using the ontology as a knowledge base. Semi-automatic and automatic (Amardeilh et al., 2005) (Alani et al., 2003) Ontology Population (OP) approaches emerged as a new field of application for knowledge acquisition techniques.

  Many approaches have been proposed. They are divided into supervised (Fleischman and Hovy, 2002), weakly supervised (Tanev and Magnini, 2006) and unsupervised (Cimiano and Völker, 2005) methods. However, ontology population is usually intended to extract information about instances from large and heterogeneous sources such as the Web. It is typically an unsupervised approach because supervised approaches need the manual construction of a training set which is not feasible for large-scale applications. These approaches use either patterns or contextual features to perform OP. Pattern-based methods seek phrases or sentence structures that explicitly show relations between instances (Mori et al., 2006). Context-based methods use a corpus[6] to extract features from the context in which a semantic class tends to appear (Tanev and Magnini, 2006).

- **Ontology enrichment process** consists in adding (removing) concepts, properties and (or) relations in the ontology or making some modifications in the already existing ones because of changes required in the ontology definition itself, and then populate it for its instances. Ontology enrichment techniques are automatic processes, which generate a set of possible modification on the evolving ontology and propose these suggestions to the ontology engineers.

  Various kinds of extensions can be made (Faatz and Steinmetz, 2004):

  - New concepts that emerge and have to be integrated. This is the most common change. Concepts may be simple or aggregate. A simple concept is a concept in the domain ontology, it explains an object in term

[6] A corpus is a large collection of documents in a specific domain providing the basis for a study.

of metadata, while, an aggregate concept is also a concept in the domain ontology that is an aggregation of simple concepts (Castano et al., 2006).

- New relations to be instantiated between existing concepts.
- Corrections of existing concepts and relations, which means that concepts and/or relations are already present in the ontology but their properties need to be rectified.

The process of ontology evolution is divided into several subtasks (Khattak et al., 2009), recapitulated in table 2.1.

Subtask	**Description**
Change request	Known as the capturing phase, identifies all the ontology modifications to apply.
Change representation	Represents the required changes, found in the previous step, using a suitable format.
Semantics change	Evaluates the effects of the changes.
Change implementation and verification	Implements the changes in ontology and then verifies that the requested changes have been committed to the ontology.
Change propagation	Ensures the propagation of changes to all dependent artifacts (applications, other ontologies,...)

Table 2.1: The process of ontology evolution

In the literature, there are approaches that consider the ontology as a whole as the evolving object (Flouris et al., 2006b; Stojanovic et al., 2002) and there are other approaches that focus on special cases such as concepts evolution (Castano et al., 2006).A summary of ontology evolution approaches is found in (Khattak et al., 2010). The main purpose of all these approaches is to provide an automation for the ontology evolution process.

2.3.2 Reasoning with Ontologies

By reasoning we mean deriving facts that are not expressed in the ontology explicitly. In fact, the emergence of ontology languages based on description logics, including OWL, allowed to enhance ontology reasoning abilities.

Several complex reasoning mechanisms are associated with ontologies, one key reasoning task is the check of (un)satisfiability by the detection of potential modeling errors such as inconsistent class descriptions and missing sub-class relationships.

Query answering is another reasoning task that is mainly used during ontology-based information retrieval. Within the most used reasoners we can mention Pellet[7] and FaCT++[8].

Reasoning in DL relies on the open world assumption. Unlike the closed world assumption, which states that *any statement that is not known to be true is false*, the open world assumption states that *everything we don't know is undefined*. That is, we cannot assume that something does not exist until it is explicitly stated that it does not exist. This has a significant impact in the way of making inferences in description logic since it makes the evidence complex to establish.

Example 2.3.1. *Let's reconsider the example of figure 2.3, James is the father of Paul. We know that James has a child. But can we infer that he has only one child? The answer depends on the assumption considered:*

- *With the closed world assumption, James has only one child as no other child is registered.*
- *With the open world assumption, we cannot exclude that James had other children even if this is not specified.*

Ontologies use only logical reasoning and do not support uncertainty. Such a hypothesis is not realistic in practice since incomplete, partial or uncertain knowledge exist in almost every aspect of ontology engineering. Hence, several extensions have been proposed to overcome this limitation, and many formalisms have been applied, among them: belief theory (Essaid and Ben Yaghlane, 2009), fuzzy logic (Stoilos et al., 2005), rough set theory (Keet, 2010) and probability theory (Ding and Peng, 2004; Yang and Calmet, 2005). The following section deals with extensions based on probability theory.

2.4 Main bridges between PGMs and ontologies

As two frameworks of knowledge representation, PGMs and ontologies can be used together in knowledge management systems in order to organize, store, and efficiently exploit information coming from different sources. We can refer to (Ribino et al., 2009) as an example of such use, where ontologies are used to organize knowledge and ontology's instances are used as input to a knowledge management module based on decision networks. However, other of works have been proposed in order to merge these two tools in such a way to take advantages of both. In what follows we will focus on this family of research.

[7] http://clarkparsia.com/pellet/
[8] http://owl.man.ac.uk/factplusplus/

2.4.1 The interoperability between PGMs and ontologies: State-of-the-art

In recent years, a panoply of works have been proposed in order to combine PGMs and ontologies so that one can enrich the other. We can outline two main directions for these proposed approaches. The first aims to enhance ontologies capabilities to support probabilistic inference. While the second aims to enhance PGMs construction by integrating ontologies.

Those which combine ontologies and PGMs in order to catch uncertainty in the knowledge domain may be classified into two main directions. The first aims to extend existing ontology languages to be able to catch uncertainty in the knowledge domain. The second aims to extend ontology querying to handle uncertainty while keeping the ontology formalism intact.

Those which concern the construction of PGMs using ontologies are divided into some works designed for specific applications, and some others that offer generic solutions to handle this issue. In what follows, we present an overview on these works.

Probabilistic graphical models for enriching ontologies power of representation.

Ontologies are based on knowledge representation formalisms that do not support uncertainty. So, some research aims to create a probabilistic extension of OWL to be able to represent probabilistic ontologies. While some other use PGMs either to maintain ontologies or to benefit from the PGMs inference abilities.

BayesOWL is a probabilistic framework based on Bayesian Networks, that aims to find a way to uncertainty modeling in OWL ontology (Ding and Peng, 2004). It uses additional markups to represent probabilistic information attached to individual concepts and properties in OWL ontologies. Then, based on a set of translation rules, this probabilistic annotated ontology could be translated to a Bayesian network and conditional probability tables attached to each node could also be generated (Ding and Peng, 2005). This framework allows to deal with ontology mapping[9] and reasoning tasks related to ontologies, such as concept overlapping[10], as probabilistic inferences. Recently, a BayesOWL API is available[11].

This research effort suffers from some weaknesses. It is restricted to translate only ontology taxonomy into BNs where nodes present only two-valued random variables.

[9] Ontology mapping process aims to build a set of matches between two ontologies.

[10] concept overlapping tries to find the degree of inclusion between a concept C and a concept represented by a description e.

[11] http://www.cs.umbc.edu/ ypeng/BayesOWL/manual/index.html

OntoBayes (Yang and Calmet, 2005) allows, as BayesOWL, uncertain knowledge representation. It inspires from BayesOWL for what is the probabilistic annotation of OWL and it adds other markups to represent dependency relations. That facilitates the explicit specification of Bayesian random variables in ontologies and allows the construction of BNs based on this specification. In addition, the Bayesian extension is implemented under OWL DL. Thus, reasoning support as well as consistency are guaranteed. Moreover, unlike BayesOWL, it allows to work with heavyweight ontologies and multi-valued random variables. Translation rules used are quite different from those used by BayesOWL.

PR-OWL (Costa and Laskey, 2006) differs from the two previous ones in the fact that it uses Multi-Entity Bayesian Networks (Laskey, 2008) (MEBN) instead of BNs to express probabilistic knowledge through OWL. MEBN is a first-order Bayesian logic that integrates classical first-order logic with probability theory. Thus this approach combines the expressive power of OWL with the flexibility and inferential power of Bayesian logic. It serves as a tool that allows probabilistic inference within an ontology language. However, some probabilistic query might be intractable or even undecidable as PR-OWL is based on the OWL FULL.

Another alternative is presented in (McGarry et al., 2007). This work was motivated by the fact that ontologies were updated by reading the research literature and then making hand annotations. In addition, in some domains, knowledge changes frequently. Consequently, the ontology updating and maintaining seems to be expensive. So, in the biological domain, they proposed to generate an ontology using information extraction techniques to catch concepts and relations between them from the research literature. This ontology is then expressed within a BN to infer and predict new protein functions.

Another approach was proposed by (Colace et al., 2003) in the specific field of intelligent e-learning systems. the authors use Bayesian networks to learn ontologies. They especially focus on inferring possible relationships among different concepts of the ontology to construct and this is via the use of BN capabilities to represent conditional dependencies among nodes.

Construction of Probabilistic graphical models based on ontologies.

Probabilistic graphical models are general frameworks for knowledge representation and inference under uncertainty. Their construction is a hard and time consuming process. Thus several approaches have been proposed to generate PGMs using ontologies. In this direction, BNs are the most commonly concerned PGMs.

The authors in (Fenz et al., 2009) provide an approach for the semi-automatic creation of BNs using existing domain ontologies that do not contain specific extensions. They use ontology relevant concepts to create nodes, ontology relations to link these nodes and use the ontological knowledge base to construct CPTs

using a mathematical function to describe dependency relationships.

To encode uncertainty in Clinical Practices Guidelines (CPG), Zheng et al. (Zheng et al., 2008) did not use existing ontologies but they created a formal model of CPG ontology that contains uncertainty features to be used to generate the DAG as well as the CPTs of the BN. The latter is then used as a tool to help doctors to judge the risk of uncertain knowledge in the clinical process using probabilistic inference.

Other researchers used ontologies to build the BN structure (Helsper and Van Der Gaag, 2002). Within the domain of oesophageal cancer, they built an ontology from a previous version of a BN and other available knowledge in the domain of interest. This ontology was then used to derive a new graphical structure of a BN through steps that focus on improving and optimizing to final resulting structure. They did not specify a method to quantify this structure as their ontology did not support uncertain informations. To overcome this limitation, they proposed the use of available probability elicitation approaches.

Devitt et al. (Devitt et al., 2006) used the inference capabilities of ontologies to automate the construction of Bayesian networks from ontologies and validated their approach in the telecommunication network management domain. This method allows even the building of dynamic Bayesian networks. To prepare the construction of the BN, concepts of interest in the domain ontology are extracted into a BN ontology.

Ben Messaoud et al. (Ben Messaoud et al., 2009) proposed the Semantical Causal Discovery (SemCaDo)algorithm. Based on MyCado algorithm, this approach uses ontological knowledge as input to learn the structure of causal Bayesian networks. In fact, this algorithm conserves same steps as MyCaDo algorithm, to which a new step of ontological knowledge integration is added in order to guide the intervention choice. It uses a utility function based on semantical inertia calculation. Unlike the other approaches described previously, this one is only interested in causal discovery and the use of ontologies aims to improve this process. In (Ben Messaoud et al., 2011), they intended to improve their approach in order to establish a real cooperation in both directions between CBNs and ontologies and this is by benefiting from the causal discovery to make the ontology evolve.

2.4.2 Summary and discussion

Ontologies provide a support for logical reasoning, but they do not support uncertainty. Hence, several extensions have been proposed to overcome this limitation. most of them use additional markups to represent probabilistic information attached to individual concepts and properties in OWL ontologies. The result is then a probabilistic annotated ontology that can be translated into a PGM to perform probabilistic inference. While, some few other works define transition ac-

tions in order to generate a PGM given an ontology with the intention of extending ontology querying to handle uncertainty while preserving the ontology formalism intact.

On the other hand, most of works that intend to construct PGMs using ontologies maintain the same mapping aspect, and they focus especially on BNs. We can recapitulate their common points as follows:

- The qualitative component
  - Concepts are associated to nodes.
  - Relations are associated to links.
  - Axioms may be useful to create either nodes or edges: For instance, in (Ding and Peng, 2004) some axioms are translated into nodes. We can also use domain restriction axioms to define the states of variables, subsumption axioms to link some nodes, etc.
- The quantitative component

  probability distributions over variables are provided by exploiting the assertional part.

Even though they represent two different paradigms, PGMs and ontologies share several issues in both representation and algorithmic aspects, we recapitulate all these similarities in table 2.2. The first part of this table recapitulates the main representational correspondences gathered from the state-of-the-art methods discussed above. While, the second part draws a parallel between the algorithmic issues related to PGMs and some of those related to ontologies. This parallelism is also based on the proposed methods and their concerns when dealing with these paradigms.

In fact ontologies allow logical reasoning, whereas, PGMs allow probabilistic one. We have seen that some works use the ontological knowledge base (its assertional part) in order to construct the probability tables of the PGM variables. The enrichment of this knowledge base is provided by the ontology population process. Furthermore, both of them require a construction phase.

Even if there were several approaches that have tried to make bridges between these two paradigms, these methods did not exploit all the ontologies capabilities as they are limited to a restrained range of PGMs, especially BNs. Consequently, they neglect some ontology important aspects such as representing concepts having more than one property, non taxonomic relations, etc. In addition, mostly, they focus on enhancing the reasoning capabilities and do not exploit other algorithmic aspects related to these paradigms.

Ontologies	**PGMs**
Representation	
Classes $\mathcal{C}$	Variables V
Relations $\mathcal{R}$	Edges E
Axioms $\mathcal{A}$	parts of E and V
Additional markups	$CPTs$
Algorithms	
Logical reasoning	Probabilistic and causal inference
Ontology population	Knowledge acquisition
Ontology building	Structure learning

Table 2.2: Table of correspondence between ontologies and PGMs

2.5 Conclusion

Ontologies are the key concept in semantic technology whose use is increasingly prevalent by the computer science community. They provide a body of structured knowledge characterized by its semantic richness and allow logical reasoning about concepts within a knowledge domain.

In this chapter, we defined ontologies, we were interested in the building task and we showed briefly some ontology reasoning matters. We have also introduced some research directions aiming to combine PGMs and ontologies.

We have seen that learning the structure of a PGM and automatic ontology enrichment are very hard problems. In the remainder of this dissertation, we will capitalize on analyzing the elements that are common to both tasks, with the intension of improving their state-of-the-art methods. The two next chapters are devoted to the description of our new two-way approach which allows PGMs and ontologies cooperation.

Chapter 3

Ontology-based generation of Object Oriented Bayesian Networks

3.1 Introduction

As we have seen in the two first chapters, PGMs suffer from their building phase known to be an NP-hard problem which can limit in some extent their application. Ontologies, from their side, provide a body of structured knowledge characterized by its semantic richness whose maintenance phase is becoming more and more crucial to ensure its continual refinement. In this work, we propose a generic gateway between these two paradigms in a two-way approach that allows PGMs and ontologies cooperation. More precisely, we propose to harness ontologies representation capabilities in order to enrich the building process of PGMs. We are in particular interested in object oriented Bayesian networks. We first generate a prior OOBN by morphing an ontology related to the problem under study and then, we describe how the learning process carried out with the OOBN might be a potential solution to enrich the ontology used initially. This chapter is devoted to describing the morphing process, while the next one focuses on the ontology enrichment process.

3.2 OOBNs vs ontologies

Unlike other studies that have focused on the use of standard BNs, we propose the use of the OOBN framework. In fact, OOBNs and ontologies share several similarities. In this section, we define the common points and similarities between these two paradigms.

3.2.1 Concepts vs classes

Ontology concepts are translated into classes of the OOBN framework. Hence, for each class so defined, concept properties will constitute the set of its random variables (real nodes). It is clear that the set of concept properties does not cover the three sets of nodes that define a class. Let:

- cp_i be the concept of the ontology translated to the class c_i in the underlying OOBN, where c_i is a DAG over $\mathcal{I}_c, \mathcal{H}_c$ and $\mathcal{O}_c$.
- $P_i = \{p_1 \dots p_k\}$ be the set of properties of cp_i.
- $\mathcal{H}'_c = \mathcal{H}_c \setminus \mathcal{H}_c^{inst}$, where $\mathcal{H}_c^{inst}$ is the set of internal nodes which are instantiations of other classes.
- $\mathcal{O}'_c = \mathcal{O}_c \setminus \mathcal{O}_c^{ref}$, where $\mathcal{O}_c^{ref}$ is the set of output nodes which are reference nodes.

P_i allows us to generate $\mathcal{H}'_c \cup \mathcal{O}'_c$. Reference nodes, namely, $\mathcal{I}_c \cup \mathcal{O}_c^{ref}$, are pointers to nodes defined in other classes. Consequently their set of states as well as parameters are copied from the referenced nodes. These latter are properties of other concepts in the ontology side. Reference nodes as well as $\mathcal{H}_c^{inst}$ will be derived from the semantic relations.

3.2.2 Inheritance relations vs class hierarchy

As ontological inheritance relations already model a hierarchical feature, then all concepts connected by an *is-a* relation in the ontology will be represented by a class hierarchy in the OOBN framework.

3.2.3 Semantic relations vs links

Having two concepts $\{cp_i, cp_j\} \in Cp^2$ related by a semantic relation means that there is at least one property of one of them that affects at least one property of the other, which means that the definition of one of them depends on the existence of the other. In the underlying OOBN, this allows to set up dependencies among nodes from different classes. Suppose that the property p_k of concept cp_i affects the property $p_{k'}$ of concept cp_j. Then, the node that represents p_k in the class c_i will be either an an output node connected directly using directed links to the internal node representing $p_{k'}$ in the class c_j, in this case c_j could only be an encapsulating class of the instance of c_i, or an output node referenced, using reference links, by a reference node in the class c_j, this reference node is either an input node,

parent of the node that represents $p_{k'}$ in c_j or an output reference node of the class containing an instance of c_i and communicates with c_j.

Semantic relations might provide an information about classes interfaces and instantiations organization in the OOBN. However, the link direction of the semantic relation can not provide a good informer about dependence relations among the variables of the OOBN, which variable depends on the other? So, it is required that the semantic relations be designed from the beginning of a causal or an anti-causal orientations. The choice of a fixed orientation is a determining factor to specify which instantiation I_i could be referenced from an instantiation I_j. Suppose that all semantic relations are of causal orientation, the cause is then conceived as the direct explanation of the fact and it is involved in its production. consequently, the definition of the concept range depends on the existence of the concept domain. In the OOBN side, this means that the definition of the class representing the concept domain is part of the class representing the concept range. This can be translated in the OOBN by instantiating the class representing the concept domain within the specification of the class representing the concept range.

In what follows, we assume that all semantic relations have an anti-causal orientation. Thus, $\forall \{cp_i, cp_j\} \in Cp^2$ related by a semantic relation, where cp_i is the domain and cp_j is the range, cp_j is considered as the cause of cp_i and this latter is the effect.

If the requiring to have all semantic relations to be anti-causal may be a hard burden on the ontologist. In this case, he could declare all the semantic relations as causal ones, and the whole process could be done in the opposite direction of the arrows.

3.3 The morphing process

To ensure the morphing process, we need to traverse the whole ontology. To provide this, we assume that the ontology is a directed graph whose nodes are the concepts and relations (semantic and hierarchical ones) are the edges. Our target is to accomplish the mapping of the ontology graphical representation into an OOBN while browsing each node once and only once. To this end, we propose to adapt the generic Depth-First Search (DFS) algorithm for graph traversing. This section is then divided into two subsections, in the first we describe the DFS algorithm, while in the second we show how to adapt it to our proposal.

3.3.1 The Depth-First Search algorithm

Many algorithms have been proposed for searching a graph. Searching a graph or graph traversal refers to the problem of following the edges of the graph, in

a particular manner, so as to visit the vertices of the graph. In this section, we focus on the depth-first search technique and we present some of its interesting properties.

The idea over the depth-first search algorithm is, as its name implies, to search "deeper" in the graph G whenever possible, G may be undirected or directed. In DFS, edges are explored out of the most recently discovered vertex v that still has unexplored edges leaving it. When all of v's edges have been explored, or there are no ones, the search backtracks to explore edges leaving the vertex from which v was discovered. Once all vertices that are reachable from the original source vertex have been discovered, the algorithm selects one of the remaining undiscovered vertices as a new source and the search is repeated from that source. This entire process is repeated until all vertices are discovered.

Whenever a vertex v is discovered during a scan of the adjacency list of an already discovered vertex u, DFS records this event by setting v's predecessor field $\pi[v]$ to u. As the search may be repeated from multiple sources, the predecessor subgraph produced by a DFS search may be composed of several trees. The predecessor subgraph of a depth-first search forms a **depth-first forest** composed of several **depth-first trees**. The edges which belong to the depth-first forest are called **tree edges**.

During the search process, the vertices are colored to indicate their state. Each vertex is initially white, is grayed when it is discovered in the search, and it is blackened when it is finished, that is, when its adjacency list has been examined completely. This technique guarantees that each vertex ends up in exactly one depth-first tree, so that these trees are disjoint.

One interesting property of DFS is that the search can be used to classify the edges of the input graph $G = (V, E)$. This edge classification can be used to glean important information about a graph. The DFS traversal allows us to define four edges types and color markers are used to keep track of which vertices have been discovered:

- **Tree edges:** are edges in the DFS forest. Edge (u, v) is a tree edge if u was first discovered by exploring edge (u, v). $\Rightarrow v$ is a white vertex.

- **Back edges:** are those edges (u, v) connecting a vertex u to an ancestor v in a DFS tree. Self-loops, which may occur in directed graphs, are considered to be back edges. $\Rightarrow v$ is a gray vertex.

- **Forward edges:** are those non-tree edges (u, v) containing a vertex u to a descendant v in a DFS tree, and u is discovered before v. $\Rightarrow v$ is a black vertex.

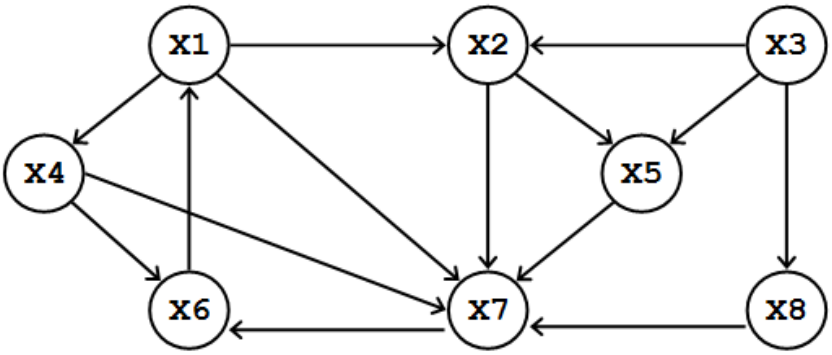


Figure 3.1: An example of a directed graph

- **Cross edges:** are all other edges. They can go between vertices in the same tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different DFS trees. $\Rightarrow v$ is a black vertex.

Example 3.3.1. *We illustrate the progress of DFS on the directed graph shown in figure 3.1. Initially, all vertices are whites.* $X1$ *is the source vertex, it is grayed. Then each vertex adjacent to* $X1$ *is recursively visited if it is white. When all vertices that are reachable from* $X1$ *have been discovered, we select* $X3$ *as a new source and we repeat the search. The algorithm finishes when all vertices have been discovered.*

The color markers are helpful in reasoning about the behavior of DFS. In the following we state the White-path theorem which gives a characterization of when one vertex is a descendant of another in the depth-first forest, and the key lemma characterizing directed acyclic graphs (Cormen et al., 2001).

Theorem 3.3.1. ***(White-path theorem)***
In a depth-first forest of a graph G*, vertex* v *is a descendant of vertex* u *if and only if at the time* $d[u]$ *that the search discovers* u*, vertex* v *can be reached from* u *along a path consisting entirely of white vertices.*

Lemma 3.3.1. G *is a directed acyclic graph if and only if a depth-first search of* G *yields no back edges.*

$\Rightarrow$ ***Proof:*** Suppose that there is a back edge (u, v). Then, vertex v is an ancestor of vertex u in the depth-first forest. There is thus a path from v to u in G, and the back edge (u, v) completes a cycle.

Suppose that G contains a cycle c. We show that a depth-first search of G yields a back edge. Let v be the first vertex to be discovered in c, and let (u, v)

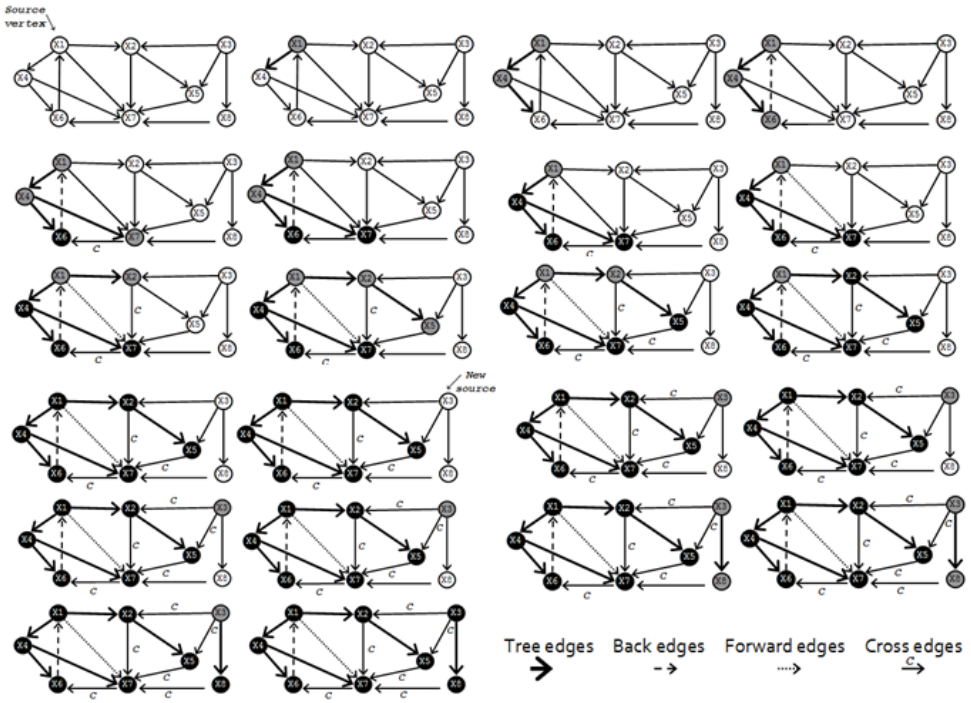


Figure 3.2: The progress of the depth-first search algorithm on a directed graph

be the preceding edge in c. Et time $d[v]$, the vertices of c form a path of white vertices from v to u. By the white-path theorem, vertex u becomes a descendant of v in the depth-forst forest. Therefore, (u, v) is a back edge.

3.3.2 The main steps of the morphing process

Our morphing process is based on the depth-first search algorithm behavior. We will use edges types to determine actions to do on each encountered concept:

- Tree edges allows to define action on concepts encountered for the first time.
- Forward and cross edges allow to define actions to do on concepts that are already visited crossing another path and so having more than one parent.
- Back edges allow cycle detection, in our case these edges will never be encountered. As our edges respect a causal orientation having a cycle of the form $X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_1$ means that X_1 is the cause of X_2 which is the cause of X_3 so this latter can't be the cause of X_1 at the same instant t but rather at an instant $t + \epsilon$. We are limited to ontologies that do not contain cycles, because such relationships invoke the dynamic aspect which is not considered in this work.

A deep study of the similarities discussed above (see section 3.2) shows that the morphing process can be done in three main steps, namely initialization, discovery and closing. At each step, we define a set of actions that might be done:

Initialization step.

All concepts are undiscovered, we generate the OOBN class and a class to each concept:

- CREATE_OOBN_GLOBAL: creates the OOBN class, meaning an environment for the classes to be instantiated in.
- CREATE_CLASS(Concept cp): transforms a concept cp to a class c_{cp}. At this step, each c_{cp} is an empty class, its sets of nodes will be defined during the other steps.

Discovery step.

We will traverse the ontology using the depth-first search algorithm, and the edges-types will be used to determine actions to do on each encountered concept. These actions allow us to define input, internal and output sets for each class of the OOBN.

- ADD_INPUT_NODE(Node n, Class c): adds an input node n to a class c. this action is invoked on all properties of a concept which is related by an out edge to another one. Its properties are considered as candidate input nodes of the class representing the second concept.
- ADD_INTERNAL_NODE(Node n, Class c): adds an internal node n to a class c. The set of internal nodes of a class consists of instantiations of other classes representing concepts that are related by an out edge to the corresponding concept of the class c in the ontology and this edge is in the same DFS search tree.
- ADD_OUTPUT_NODE(Node n, Class c): adds an output node n to a class c. all properties of a concept are transformed into variables of its corresponding class in the OOBN. These nodes are considered as candidate output nodes of the class.
- ADD_OUTREF_NODE(Class $c1$, Class $c2$): adds output reference nodes to classes containing $c1$ until reaching $c2$. In fact, some concepts might have parents coming from more than one DFS search tree. Let cp_i be a concept having two parents $cp1_i$ and $cp2_i$ coming from two different branches. Then,

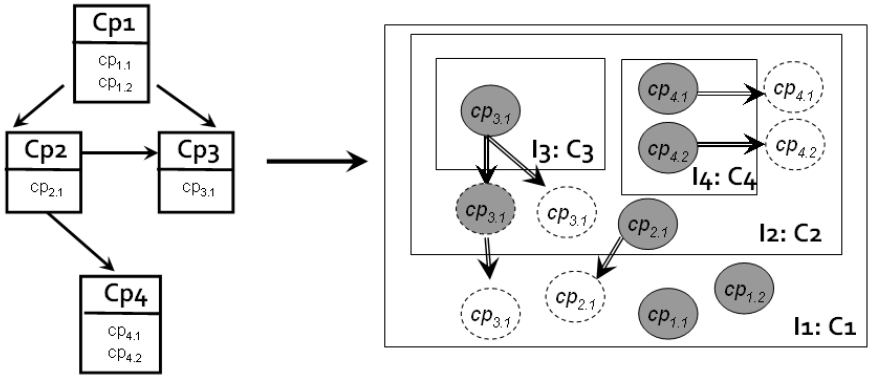


Figure 3.3: An example of more than one parent

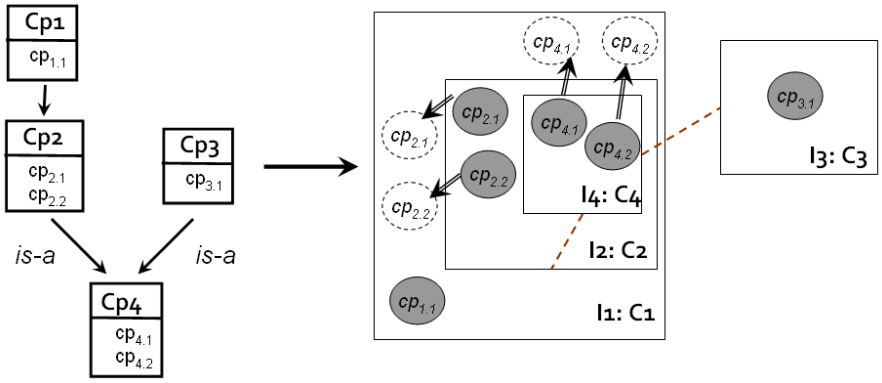


Figure 3.4: An example of inheritance relation

c_{cp_i} would to be instantiated within the specification of only one of them. However, c_{cp_i} has its output nodes to be referenced by output reference nodes of the class containing it until reaching its second parent (see figure 3.3).

- ADD_CONSTRUCT_LINK(Class $c1$, Class $c2$): a construct link appears between instantiations of superclasses and instantiations of their subclasses (see figure 3.4). All properties of the super-concept are considered as properties of its subconcepts.

- ADD_REFERENCE_LINK(Node $n1$, Node $n2$): allows the communication between classes interfaces.

Closing step.

We check whether the vertex is a root (having no predecessor), if it is, we add an instance of its class to the global OOBN class using the ADD_INTERNAL_NODE

Algorithm 3: Onto2PriorOOBN

Input: An ontology $\mathcal{O}$.

For all concepts, the color must be initialized to "white" before running the algorithm.

```
begin
    CREATE_OOBN_GLOBAL ;
    for each concept cp ∈ Cp do
        RECORD_PREDECESSOR[cp]=NULL;
        CREATE_CLASS(cp)
    for each concept cp ∈ Cp do
        if color[cp]= white then
            Handling_Process(O, cp)
```

action.

We also define the INSTANCE_OF (Class c) which allow to instantiate a class c and the GET_OUTPUT(Class c) which returns all output nodes of a class c.

All these actions are used in Algorithms 3 and 4. The *Handling_ Process* function (see algorithm 4) provides actions to do at each vertex. In fact, we generate the OOBN at the moment of the ontology exploration.

Example 3.3.2. *We assume that all random variables related to the problem under study are modeled in the corresponding ontology as concepts properties and that all semantic relations present in the ontology are of an anti-causal orientation. By applying the Onto2PriorOOBN algorithm to the ontology of figure 3.5.(a), and with respect to the correspondences discussed above between OOBNs and ontologies, figure 3.5.(b) represents the prior OOBN that can be generated.*
In the initialization step, we have created the Prior OOBN class as well as classes representing concepts.
The discovery step allowed us to determine the sets of nodes of each class.
Finally, the closing step allowed to instantiate the classes c_{cp1}, c_{cp3} and c_{cp4} (represented respectively by I_1, I_3 and $I4$ in figure 3.5.(b))in the Prior OOBN class.

3.4 The learning process

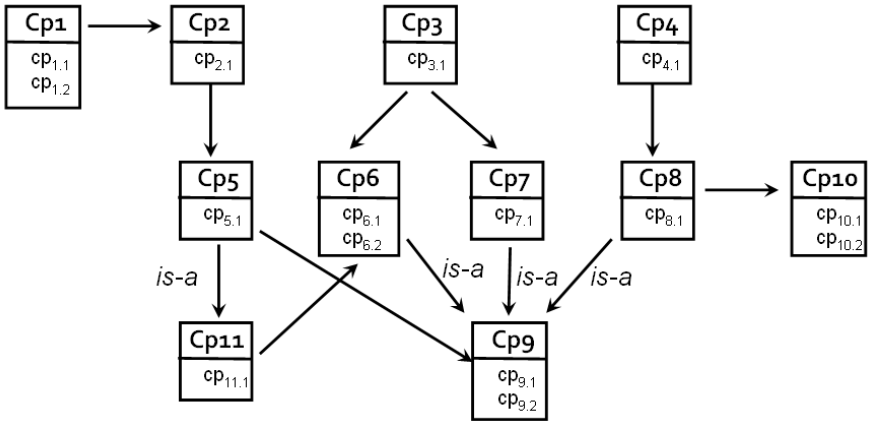
To ensure the learning process, we adopt the OO-SEM algorithm (see algorithm 1) described in the first chapter. We remind that this algorithm is based on two phases: The first takes advantages from prior information available when learning in object oriented domains, by asking an expert about a partial specification of

Algorithm 4: Handling_Process

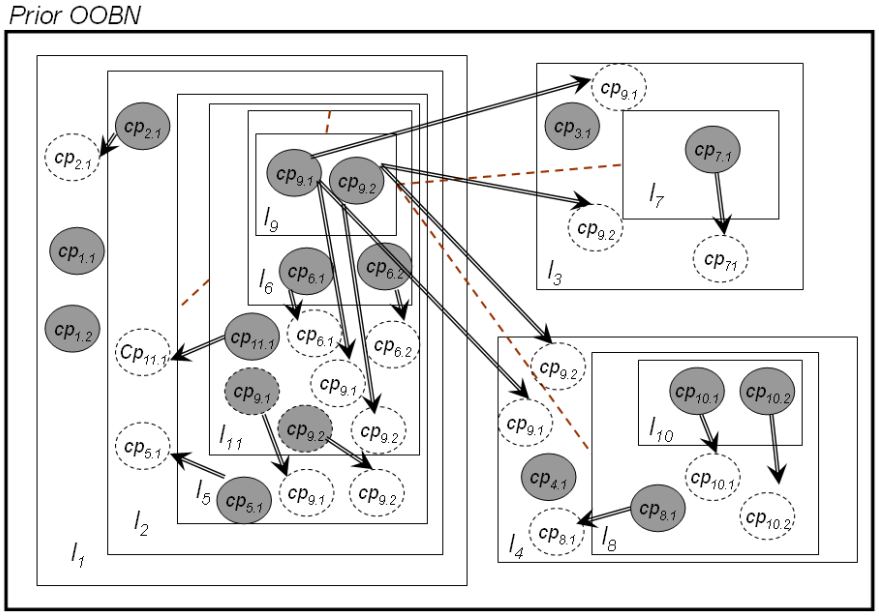
Input: An ontology $\mathcal{O}$, A concept S.

```
begin
    color[S]:= gray;
    for each property p of S do
        ADD_OUTPUT_NODE(p, c_S)
    for each V ∈ adjacent[S] do
        if color[V]=white then
            RECORD_PREDECESSOR[V]=S;
            Handling_Process(O, V);
            ADD_INTERNAL_NODE
            (INSTANCE_OF(c_V), c_S);
            if (S,V) is an inheritance relation then
                ADD_CONSTRUCT_LINK
                INSTANCE_OF(c_V),INSTANCE_OF(c_S))
            if (S,V) is a semantic relation then
                for each node n ∈ GET_OUTPUT(c_V) do
                    ADD_REFERENCE_LINK
                    (n,ADD_INPUT_NODE(n,c_S))
        if color[V]=black then
            if (S,V) is an inheritance relation then
                ADD_CONSTRUCT_LINK
                INSTANCE_OF(c_S),INSTANCE_OF(c_V))
            if (S,V) is a semantic relation then
                for each node n ∈ GET_OUTPUT(c_V) do
                    ADD_INPUT_NODE(n, c_S)
                ADD_OUTREF_NODE
                (INSTANCE_OF(c_S),INSTANCE_OF(c_V)
    color[S]:= black;
    if RECORD_PREDECESSOR[S] = Null then
        ADD_INTERNAL_NODE
        (INSTANCE_OF(c_S),GLOBAL_OOBN_CLASS)
```

an OOBN by grouping nodes into instantiations and instantiations into classes. Having this prior model, the second step adapt the SEM algorithm (Friedman, 1998) in order to learn the OOBN structure that fits best to the data. It starts by learning the interfaces of the instantiations. Then, the structure inside each



(a) The ontology to morph



(b) The resulting OOBN

Figure 3.5: An example of ontology morphing to a prior OOBN

class is learned based on the candidate interfaces founded previously. Thanks to the morphing process described above, we are not in need of expert elicitation as our process allowed us to take advantage of the semantic richness provided by ontologies to generate the prior OOBN structure. This latter is then used as a starting point to the second phase and it ensures a good start-up to the classical building process as it allows us to gather both semantical and observational data. Algorithm 5 recapitulates our learning steps.

Algorithm 5: OntoAndData2OOBN

Input: ontology $\mathcal{O}$, observational data $\mathcal{D}$
Output: OOBN
begin
  $OOBN_{prior} = Onto2PriorOOBN(\mathcal{O})$
  $OOBN = OO\text{-}SEM(OOBN_{prior}, \mathcal{D})$

3.5 Conclusion

In this chapter, we showed how we take advantage of the semantic richness provided by ontologies to generate an OOBN structure and this is by exploring similarities between these two paradigms. The use of the OOBN framework has enabled us to handle an extended range of ontologies unlike works which were limited to the use of standard Bayesian networks. In the next chapter, we continue the description of our approach and we outline the second part which describes the ontology enrichment process based on the OOBN structure learning algorithm.

Chapter 4

Ontology enrichment through OOBN structure learning

4.1 Introduction

In the previous chapter, we have represented how to generate a prior OOBN by morphing an ontology and how to use it as a starting point to the global building OOBN algorithm. In this chapter, we deal with the second part of our two-way approach and we describe how the learning process carried out with the OOBN might be a potential solution to enrich the ontology used initially.

4.2 Ontology enrichment vs structural OOBN learning

We have generated a prior OOBN by morphing an ontology , then we have used it as a starting point to the global building OOBN algorithm which learned from data the final OOBN structure. By this way, we have gathered both semantic richness and observational data. Yet, in some cases, data may contradict the ontological knowledge which leads us to distinguish two possible working assumptions:

- **A total confidence in the ontology.** Any contradiction encountered during the learning process is due to data. The conflict must be managed while remaining consistent with the ontological knowledge. The enrichment process will be restrained to the addition of certain knowledge (relations, concepts, etc.) while preserving the already existing ones and what was regarded as truth remains truth.
- **A total confidence in the data.** Any contradiction encountered during the

learning process is due to the ontology. The conflict must be managed while remaining faithful to the data. In this case, changing the conceptualization of the ontology will be allowed, not only by adding knowledge, but also by performing other possible changes, such as deleting, merging, etc. This means that the data will allow us to get new truths that may suspect the former.

In (Xuan et al., 2006), the authors make the difference between ontology evolution and revolution. The former consists in adding new truths to old ones which matches our first case. The latter consists in approving that old truths might be false which matches our second case. In the following we will not be restrained to the evolution case but we will be interested in the ontology revolution.

As described above, the OO-SEM algorithm starts by learning the interface of each class then learns the structure inside each class. We can benefit of these two steps in order to improve the ontology granularity. In fact, the first step allows us to detect the possible existence of new relations and / or concepts which might be added to the ontology at hand. The second step may affect the definition of the already existing concepts and / or relations.

4.2.1 Interfaces learning vs relations and / or concepts adding or removing

Discovering that classes c_i and c_j are related or not may be analyzed in different manners to ensure the ontology enrichment.

Remove relations

When we have generated the OOBN, concepts related by a semantic relations were represented by instances of classes encapsulated in each other, and we have expect that the learning process will identify the variables that interact between the two instances. If no common interface is identified, then these two concepts should be independent. So their semantic relation have to be checked.

Example 4.2.1. *Figure 4.1 shows an example of the remove relation rule, here we suppose that after performing the learning process, we didn't find nodes from C_{cp_1} that reference nodes from C_{cp_2}. Thus, we can propose to delete the semantic relation which appears in the ontology.*

Add concepts / relations

If c_i communicates only with c_j, then this may be translated by simply adding a new relation between concepts representing these classes in the underlying ontology. Otherwise, if c_i communicates with c_j and a set S_c of other classes, then we

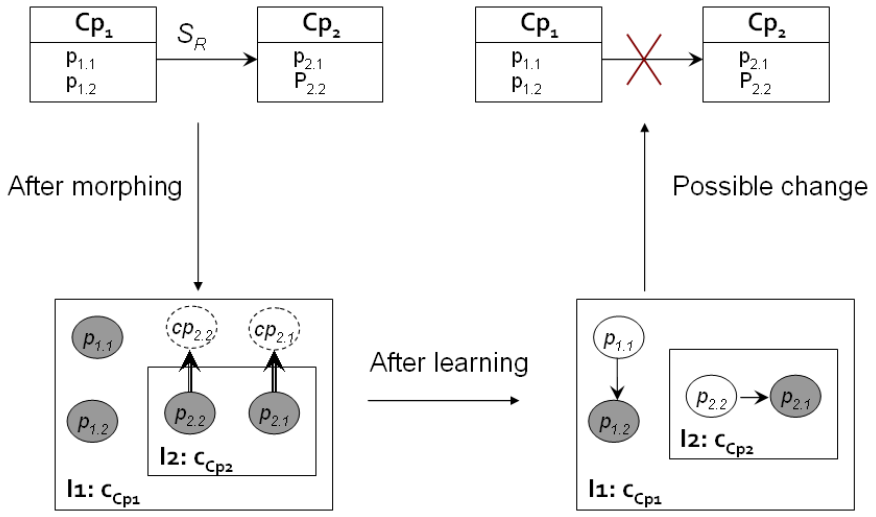


Figure 4.1: Enrichment process: an example of removing a relation (S_R is a semantic relation)

can use this exchange between classes whether by translating it into relations or concepts allowing the factorization of some other ones already present in the ontology. In fact, it would be interesting to check whether or not c_j and S_c share some similarities. If these classes share similar sets of nodes and have similar structures, then we can extract a super-concept over them. Otherwise, these relations will be also translated into relations in the ontology.

Example 4.2.2. *Figure 4.2 shows an example of the adding concept-relation rules. We suppose that $p_{2.1}$ and $p_{4.2}$ have the same state space and their respective classes C_{cp_2} and C_{cp_4} communicate with the same class C_{cp_1}. Thus, in the ontological side, we can define a super-concept over cp_2 and cp_4 having p_{super} as property which substitute both $p_{2.1}$ and $p_{4.2}$. Note that C_{cp_3} communicates also with C_{cp_1}, but as it doesn't represent shared properties with the other classes, then we will simply add a new relation between cp_3 and cp_1 in the ontology.*

4.2.2 Classes learning vs concepts redefinition

In this part, we will look into the concepts definition. As defined above, each class c in the OOBN is a DAG over its three sets of nodes $\mathcal{I}_c, \mathcal{H}_c$ and $\mathcal{O}_c$. Suppose that the result of the learning process, was a disconnected graph (as the example of figure 4.3), this means that $\mathcal{I}_c, \mathcal{H}_c$ and $\mathcal{O}_c$ are divided into multiple independent connected components (two in figure 4.3). Thus, nodes of each component are not really correlated with those of the other components. This can be translated in

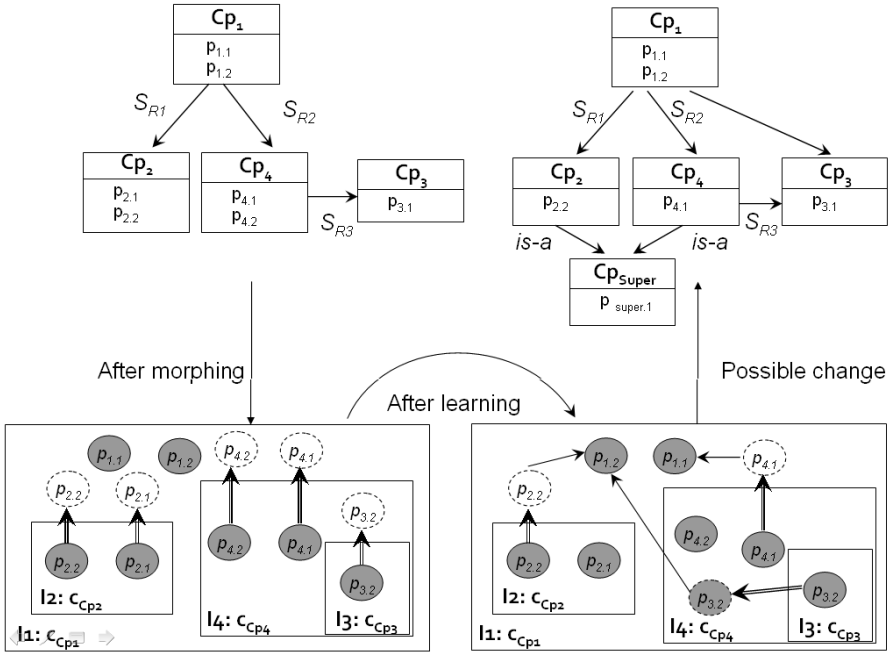


Figure 4.2: Enrichment process: an example of adding concepts and relations (S_{R1}, S_{R2} and S_{R3} are semantic relations)

the ontological side by proposing to deconstruct the corresponding concept into more refined ones, where each concept represents a component of the disconnected graph.

Example 4.2.3. *Figure 4.3 shows an example of the concept-redefinition rule. After performing the learning process, the class C_{cp} representing the concept cp consists of two components. Thus we can propose to replace the concept cp by two more refined concepts, each of them represents a component.*

The possible changes are then communicated to an expert via a warning system

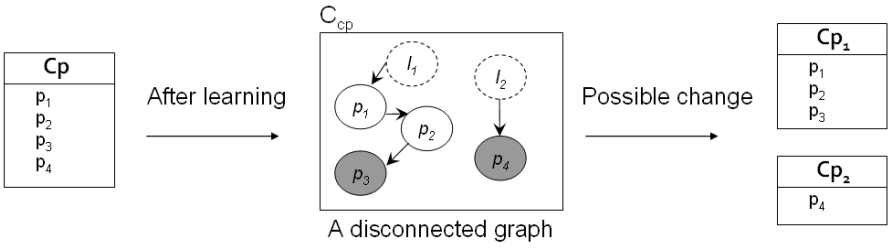


Figure 4.3: Enrichment process: an example of concept redefinition

which detects the changes and allows the expert to state actions to be done. If he chooses to apply the change, he shall first denominate the discovered relations and / or concepts.

4.3 The ontology enrichment process

As we have done for the morphing process, after discussing the set of possible changes that may be allowed thanks to the learning process, here we define a set of actions to perform having the final OOBN structure, in order to suggest the appropriate enrichment rule to the ontologist. Table 4.1 recapitulates the possible changes that we can deduce from the learning process.

Change	**Rule name**	**Description**
Remove a relation	REMOVE_RELATION (Concept *cp*)	Removes the relations that rely concept *cp* to other ones in the ontology.
Add a relation	ADD_RELATION (Concept *cp*1, Concept *cp*2)	Adds a relation between *cp*1 and *cp*2 in the ontology.
Create a super-concept	CREATE_SUPER (List of concepts)	Creates a super-concept of the parameters concepts.
Remove properties	REMOVE_SIMILAR_PROPERTIES (List of concepts)	Removes the set of similar properties from the set of parameter concepts.
Add *is-a* relation	ADD_ISA_RELATION (Concept *cp*1, List of concepts)	Adds an *is-a* relation between the concept *cp*1 and each concept in the list of concepts.
Add concept	ADD_CONCEPT (List of properties)	Creates a new concept over the parameters properties.
Remove concept	REMOVE_CONCEPT (Concept *cp*)	Removes the concept *cp* from the ontology.

Table 4.1: The possible changes

Algorithm 6: Generate_Enrichment_Rules

Input: An OOBN structure
Output: A set of suggested rules

```
begin
    for each instantiation t of a class c_cp in the OOBN do
        Related_Classes := CONNECTED_TO(t);
        if Length(Related_ Classes)=0 then
            REMOVE_RELATION(GET_CONCEPT(t));
        if Length(Related_ Classes)=1 then
            ADD_RELATION(GET_CONCEPT(t),GET_CONCEPT(Related_Classes[1]))
        if Length(Related_ Classes)>1 then
            CHECK_SIMILAR(Related_Classes);
            for each subset S of similar classes do
                if Length(S)>1 then
                    for i:=1 to Length(S) do
                        S' := S'+ GET_CONCEPT(S[i])
                    Super:=CREATE_SUPER(S');
                    REMOVE_SIMILAR_PROPERTIES(S');
                    ADD_ISA_RELATION(Super, S');
                else
                    ADD_RELATION(GET_CONCEPT(t),GET_CONCEPT(S[1]));
        Components:= COUNT_COMPONENNT;
        if Components>1 then
            for each Component comp of t do
                for each real node v ∈ comp do
                    List_Properties := List_Properties + GET_PROPERTY(v)
                for each reference node u ∈ comp do
                    List_Related := List_Related +
                    GET_CONCEPT(GET_CLASS(u))
                New_Concept:= ADD_CONCEPT(List_Properties);
                for each concept cp' ∈ List_ Related do
                    ADD_RELATION(New_Concept, cp');
            REMOVE_CONCEPT(GET_CONCEPT(t));
            REMOVE_RELATION(GET_CONCEPT(t));
```

We also define a set of functions allowing the OOBN-Ontology communication. These functions are useful in algorithm 6 to specify, as it may be the case, the change that should be invoked.

- CONNECTED_TO(Class T): gives the set of classes that class T communicates with via its interface.
- GET_CONCEPT(Class T): returns the concept cp_T of the ontology corresponding to the class T in the OOBN.
- GET_CLASS(Node n): returns the class of the node n in the OOBN.
- GET_PROPERTY(Node n): returns the concept property corresponding to the node n in the OOBN.
- CHECK_SIMILAR(List of concepts): gathers similar concepts in a same list. It returns a list containing the lists of similar classes.
- COUNT_COMPONENT: returns the component number in an OOBN class.

4.4 Our new OOBN-Ontology Cooperation (2OC) approach

In the previous and current chapters, we have outlined the main steps of our proposal. Here, we summarize all this steps in a new two-way approach named OOBN-Ontology Cooperation (2OC) approach, then we illustrate the entire process through a detailed example.

4.4.1 Summary

Our work was first motivated, by the increased use of PGMs as well as ontologies in real world application. Then, it was also motivated by the state-of-the-art methods proposed in order to combine these two frameworks and which were mostly interested in the use of the BN framework as this latter is largely developed and used in several real world applications. On one hand, this choice seems to be limited as it does not explore all the expressive capabilities of ontologies. On the other hand, the main idea behind these methods was to enhance ontology reasoning abilities to support uncertainty. However, we cannot provide a good basis for reasoning while we do not have a well defined ontology, which takes into consideration changes of its knowledge domain. The ontology building process is a tedious process which often requires human expertise. Thus, several works have been made in order to automate this process. Some of which are interested in the creation step while some other focus on the maintenance step and how to update the ontology given possible changes while preserving its consistence. It is within this context, that we have designed our two-way approach. The main idea behind our 2OC approach

was to use semantical knowledge already introduced in the ontology in order to help learning the structure of a probabilistic graphical model, precisely, an OOBN. In fact, object oriented Bayesian networks are an extension of standard Bayesian networks using the object paradigm and sharing several similarities with ontologies. Thus, finding a mapping process of ontologies into OOBNs will certainly allow us to deal with ontologies without significant restrictions. We have defined a process allowing the morphing of an ontology into a prior OOBN structure, this latter is then learned using observational data. The final structure derived from the learning process is the result of a mixture of semantical data provided by the ontology and observational data. This observational data may not be explored by ontologists or this mixture may allow to discover new knowledge which is not yet expressed by the ontology. Thus, our idea was to set some mapping rules allowing to translate new relations discovered by the learning process into knowledge which may be useful to enrich the initial ontology. The main steps of our proposal are sketched in figure 4.4.

We should mention that this process relies on the following set of assumptions:

- ***Constraints on the ontology***
  - **All the semantic relations should be of causal or anti-causal orientation.** This assumption is useful to facilitate the morphing process as it provides information about classes interfaces and instantiations organization in the OOBN.
  - **The ontology graph should be acyclic.** To avoid the dynamic aspect of the OOBN framework at the morphing step.
- ***Constraint on the data***
  - **A total confidence in the data.** This assumption is useful to condition the enrichment process. Setting up this assumption means that the data will allow us to get new truths that may suspect those already claimed by the ontology.

4.4.2 Practical example

Our illustration is based on the well-known insurance Bayesian network (Binder et al., 1997), taked from the Bayesian Network Repository[1]. The insurance network is designed for evaluating car insurance applications. It consists of 27 discrete variables connected by 52 directed arcs. This network was represented in (Langseth and Nielsen, 2003) using the OOBN framework as depicted in figure 4.6. The

[1] http://www.cs.huji.ac.il/site//labs/compbio/Repository/

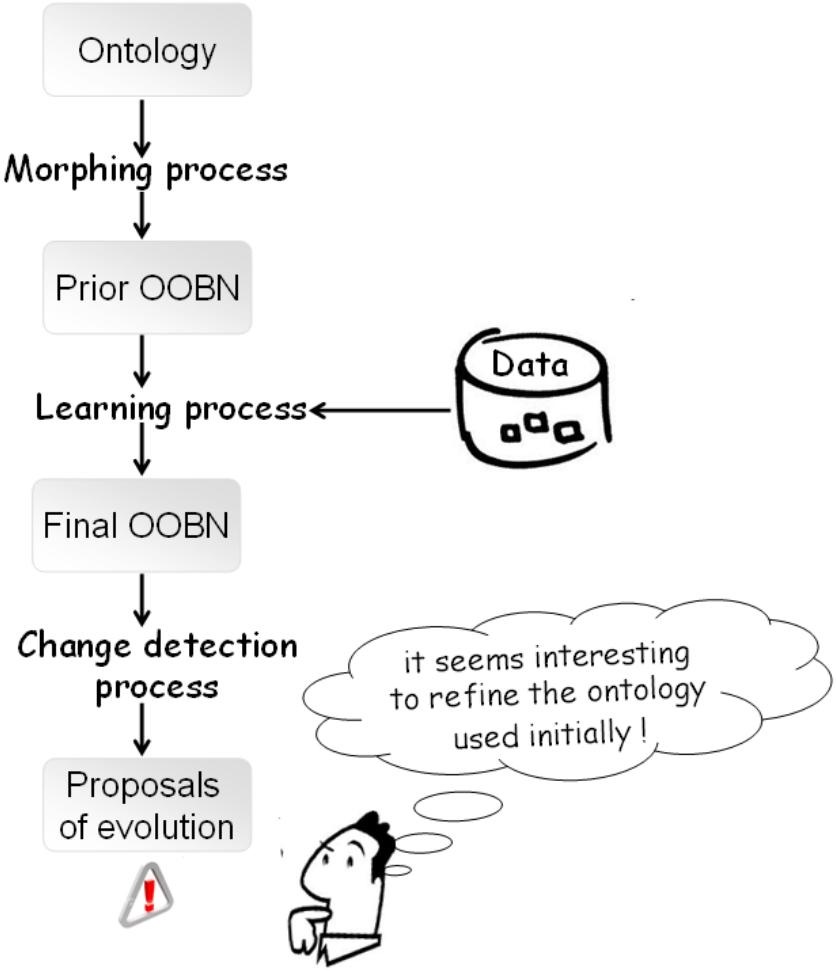


Figure 4.4: The whole process of our approach

insurance network represented using the OOBN framework contains six classes (Insurance, Theft, Accident, Car, CarOwner and Driver). In figure 4.6 only the interfaces of the encapsulated instantiations are shown, dashed ellipses represent input nodes, while shaded ellipses represent output nodes. The sets of input, internal and output nodes for each class are:

- ***Insurance:***
  - $\mathcal{I} = \{\text{Age}\}$
  - $\mathcal{H} = \{\text{MedCost, ThisCarDam, ThisCarCost, PropCost, ILiCost, OtherCarCost, CO:CarOwner, T:Theft, C:Car, A:Accident}\}$
  - $\mathcal{O} = \{\emptyset\}$

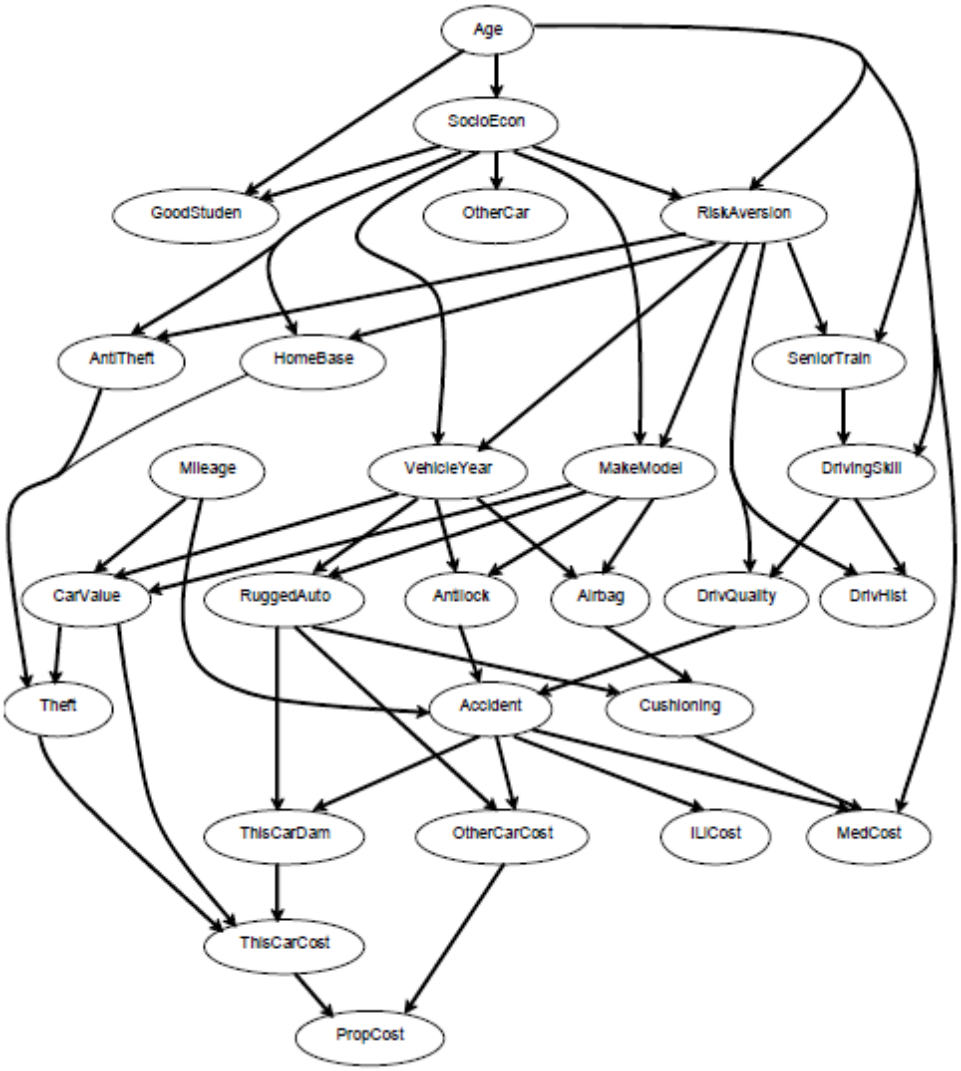


Figure 4.5: The insurance Bayesian network

- ***CarOwner:***
  - $\mathcal{I} = \{\emptyset\}$
  - $\mathcal{H} = \{$GoodStuden, OtherCar, D:Driver$\}$
  - $\mathcal{O} = \{$Age, SocioEcon, HomeBase, AntiTheft, VehicleYear, MakeModel, DrivQuality$\}$
- ***Driver:***
  - $\mathcal{I} = \{$SocioEcon, Age$\}$

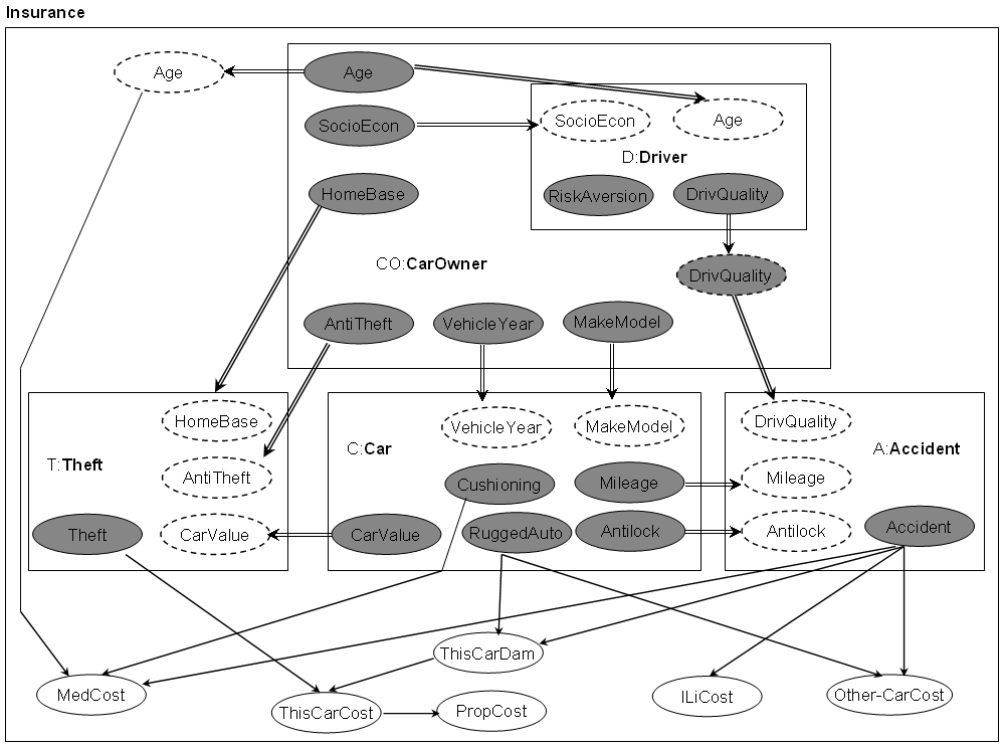


Figure 4.6: The insurance network represented using the OOBN framework

- $\mathcal{H}$ = {SeniorTrain, DrivingSkill, DrivHist}
- $\mathcal{O}$ = {RiskAversion, DrivQuality}

- ***Theft:***
  - $\mathcal{I}$ = {HomeBase, AntiTheft, CarValue}
  - $\mathcal{H}$ = {$\emptyset$}
  - $\mathcal{O}$ = { Theft }

- ***Car:***
  - $\mathcal{I}$ = {VehicleYear, MakeModel}
  - $\mathcal{H}$ = {Airbag}
  - $\mathcal{O}$ = {Cushioning, Mileage, CarValue, RuggedAuto, Antilock}

- ***Accident:***
  - $\mathcal{I}$ = {DrivQuality, Mileage, Antilock}

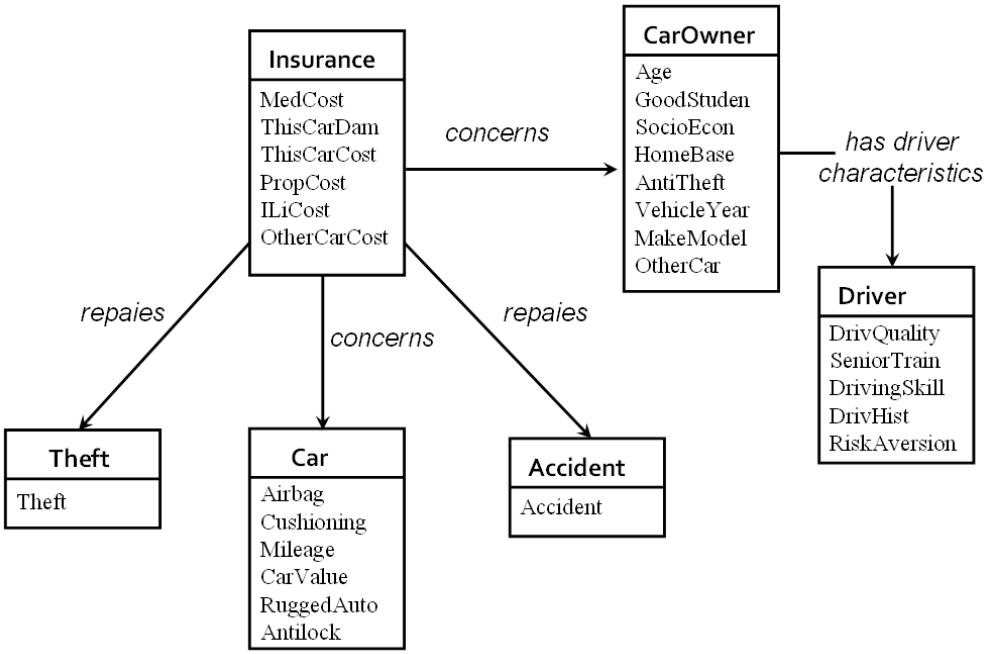


Figure 4.7: The insurance ontology

- $\mathcal{H} = \{\emptyset\}$
- $\mathcal{O} = \{\text{Accident}\}$

Our example is organized as follows:

1. We will generate an insurance ontology containing all the variables of the insurance network.

2. We will morph the insurance ontology into a prior OOBN.

3. We will assume that this prior model and observational data used to learn the structure lead to the OOBN of figure 4.6, and we will discuss how this structure may improve the ontology used initially.

The generated ontology

The Insurance ontology consists of six concepts, namely Insurance, Theft, Accident, Car, CarOwner and Driver which are related using semantic relations as represented in figure 4.7. Each concept has a set of properties, these properties represent the variables of the insurance network.

The OOBN generation process

We will follow the steps of the Onto2PriorOOBN algorithm (see algorithm 3)to generate our prior OOBN.

First of all, we start by generating the Global OOBN class *Global_ Insurance*. Then we create a class to each concept of the ontology, that is, we create six classes Insurance, Theft, Accident, Car, CarOwner and Driver which are initially empty. their sets of nodes will be discovered during the generation process.

Initially, all concepts are white. *Insurance* is the source concept, it is grayed. Then, each concept adjacent to *Insurance* is recursively visited if it is white. *Insurance* has *CarOwner, Theft, Car* and *Accident* as adjacent concepts. We start by discovering *CarOwner*, it is painted gray and it has *Driver* as adjacent concept. *Driver* has no adjacent, all its properties are declared as output nodes of the class representing it in the prior OOBN and it is instantiated within its ancestor *CarOwner*. As *CarOwner* and *Driver* are related by a semantic relation then, all its output nodes are considered as input nodes of the *CarOwner* class linked to them using reference links. the concept *Driver* is finished and blackened. We backtrack to the *CarOwner* concept, it is gray and it has finished his adjacent concepts so, all its properties are declared as output nodes of the class representing it in the prior OOBN and it is instantiated within its ancestor *Insurance*. As *Insurance* and *CarOwner* are related by a semantic relation then, all its output nodes are considered as input nodes of the *Insurance* class linked to them using reference links. The *CarOwner* is blackened and we go back to the second adjacent of the concept *Insurance* and so on until discovering all the concepts. The result of this process is shown in figure 4.8.

The Learning process

The prior OOBN generated above allows us to get the OOBN classes organization and gives us a set of candidate interfaces. This latter is not exhaustive thus, the learning process starts by learning the classes interfaces before learning the structure inside each class. We assume that the result of the learning process is the OOBN depicted in figure 4.6.

The ontology enrichment process

The OOBN of figure 4.6 represents new relations between class instances which were not present in the prior model. As we have only the interfaces we assume that the structure inside each class is a connected DAG. We will run the Generate_Enrichment_Rules algorithm (see algorithm 6) to discover which new knowledge may be added to the insurance ontology.

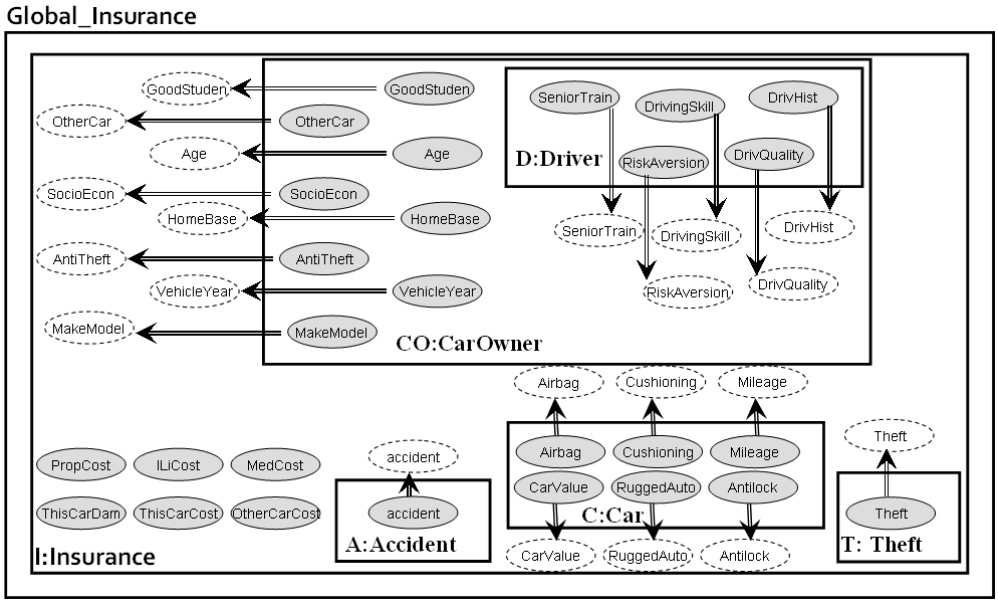


Figure 4.8: The prior OOBN of the insurance ontology

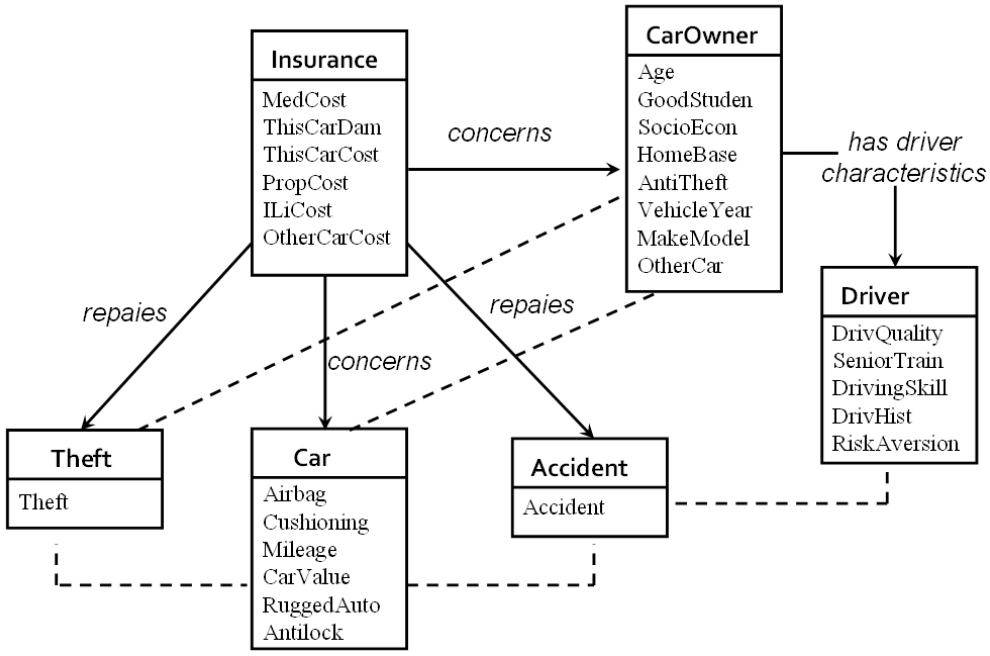


Figure 4.9: Possible changes on the insurance ontology

The class *Insurance* communicates with the classes *CarOwner, Theft, Car* and *Accident*. This was already expressed by the prior OOBN and it is affirmed by

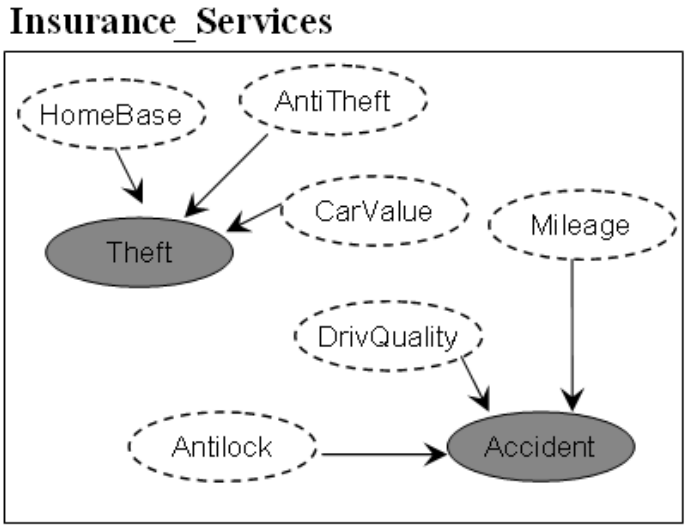


Figure 4.10: The Insurance_Services class

the final structure. The algorithm will check similarities between this classes, all of them are different so, no changes are required.

Nevertheless, if we suppose that the output node of the class *Theft* was not linked to the internal node *ThisCarCost* of the *Insurance* class then, according to our process, we will suggest to the ontologist to remove the relation between the *Insurance* concept and the *Theft* concept.

The class *CarOwner* communicates with the classes *Driver, Theft, Car* and *Accident*. This was not expressed by the prior OOBN, except for the class *Driver*. The algorithm will check similarities between this classes, all of them are different so, we will simply propose to add a relation between the *Insurance* concept and each concept corresponding to the *Theft, Car* and *Accident* classes. Similarly, the algorithm will deal with the classes *Driver, Theft, Car* and *Accident*. Changes that may be made to the ontology are dotted in figure 4.9.

To illustrate the case where we can deconstruct a concept into more refined ones, we assume that the concepts *Theft* and *Accident* are represented by just one concept named *Insurance_Services* having Theft and Accident as properties. After the learning process, we will have the *Insurance_Services* class containing two components as shown in figure 4.10. This concept may then be removed and replaced by two concepts, each of them corresponds to one component.

4.5 Conclusion

In this chapter, we dealt with the second part of our approach, we carefully described the main enrichment actions that may be deduced from the learning process and we gave a generic algorithm to describe how to apply these actions. Then, we recapitulated the main phases of our two-way approach while reminding the working assumptions. Finally, we gave a detailed illustration of the whole process.

Conclusion

In this master thesis, we have proposed a new two-way approach, named OOBN-Ontology Cooperation (2OC) for OOBN structure learning and automatic ontology enrichment. The leading idea of our approach is to capitalize on analyzing the elements that are common to both tasks with the intension of improving their state-of-the-art methods.

Roughly speaking, our 2OC approach includes two interdependent parts. The first consists in morphing an ontology related to the problem under study into a prior OOBN in order to use it as a starting point to the global OOBN building algorithm. By this way, the classical learning process will take advantage from semantical data derived from ontology while minimizing expert involvement. The second uses the final structure resulting from the learning process to enrich the ontology used initially by interpreting new relations discovered and setting up a set of mapping rules allowing to translate these relations into new knowledge to integrate in the ontology.

In fact, our work is considered as an initiative aiming to set up new bridges between PGMs and ontologies as research which has been proposed in order to combine PGMs and ontologies were limited to a restrained range of PGMs, usually BNs, so they neglect some important ontology aspects. In addition, these methods were especially interested in either enhancing ontologies capabilities to support probabilistic inference or improving PGMs construction by integrating ontologies. Thus, the originality of our method lies first, on the use of the OOBN framework which allowed us to address an extended range of ontologies, second, on its bidirectional benefit as it ensures a real cooperation, in both ways, between ontologies and OOBNs.

We introduced the algorithmic aspect of our approach and we have illustrated the whole process via a detailed example. As a first line of research, we aim to implement our method and test it in real world applications based on ontologies.

We notice that the first part of our method, namely, the morphing process has been accepted for oral presentation and as a full paper to be included in the electronic proceedings of the 8th Bayesian Modeling Applications Workshop (BMAW-11), part of the Conference on Uncertainty in Artificial Intelligence (UAI). The summarized version of the OOBN-Ontology Cooperation approach has been submitted as a full paper in the International Conference on Knowledge Engineering and Ontology Development (KEOD).

Nevertheless, this current version is subject to several improvements to both the learning and the enrichment processes. As future work, we consider different tracks of refinement that we classify into three major research directions:

- We will be interested in using the semantical richness of ontologies not only to construct a prior PGM but also to help finding its final structure, and this is by proposing new metrics, based on ontological knowledge, allowing to assess better the choice of the best structure.
- Another interesting direction of research is the causal discovery for OOBN. As we have already mentioned, (Ben Messaoud et al., 2011) proposed a similar approach to ours based on causal Bayesian networks. Thus, we propose to extend the OOBN learning process by discovering causal relationships and providing *Object Oriented Causal Bayesian Networks (OOCBNs)*. By this way, our proposal will preserve the causal aspect provided (Ben Messaoud et al., 2011) and handle different ontologies without significant restrictions.
- Furthermore, in this work, our aim was to provide a warning system able to propose a set of possible changes to the ontology engineers. The discovered relations and / or concepts have to be denominated so, as a possible research direction, we will be interested in natural language processing (NLP) methods to allow the automation of this process.

Bibliography

Alani, H., Kim, S., Millard, D. E., Weal, M. J., W. Hall, P. H. L., and Shadbolt, N. R. (2003). Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems*, 18(1):14–21.

Amardeilh, F., Laublet, P., and Minel, J.-L. (2005). Annotation documentaire et peuplement d'ontologie à partir d´extractions linguistiques. In *Actes des 16èmes journée francophones d´Ingénierie des Connaissances*, pages 25–36.

Bangsø, O., Langseth, H., and Nielsen, T. D. (2001). Structural learning in object oriented domains. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*, pages 340–344, Key West, Florida, USA. AAAI Press.

Bangsø, O. and Wuillemin, P.-H. (2000a). Object oriented bayesian networks: a framework for top-down specification of large bayesian networks with repetitive structures. Technical report, Department of Computer Science, Aalborg University, Denmark.

Bangsø, O. and Wuillemin, P.-H. (2000b). Top-down construction and repetitive structures representation in bayesian networks. In *Proceedings of the Thirteenth International Florida Artificial Intelligence Research Society Conference*, pages . 282–286, Orlando, Florida, USA. AAAI Press.

Ben Messaoud, M., Leray, P., and Ben Amor, N. (2011). Semcado: a serendipitous strategy for learning causal bayesian networks using ontologies. In *Proceedings of the 11th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (to appear)*, pages ?–?, Belfast, Northern Ireland.

Ben Messaoud, M., Leray, P., and N.Ben Amor (2009). Integrating ontological knowledge for iterative causal discovery. In *Proceedings of the 10th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 168–179, Verona, Italy.

Binder, J., Koller, D., Russell, S., and Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29.

Borchani, H., Ben Amor, N., and Mellouli, K. (2006). Learning bayesian network equivalence classes from incomplete data. In *Proceedings of the Ninth International Conference on Discovery Science, Lecture Notes in Artificial Intelligence*, pages 291–295, Hammamet, Tunisia.

Borchani, H., Chaouachi, M., , and Ben Amor, N. (2007). Learning causal bayesian networks from incomplete observational data and interventions. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Hammamet, Tunisia.

Buntine, W. L. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8:195–210.

Castano, S., Ferrara, A., and Hess, G. (2006). Discovery-driven ontology evolution. In *The Semantic Web Applications and Perspactives (SWAP), 3rd Italian Semantic Web Workshop*, PISA, Italy.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554.

Chickering, D. M., Geiger, D., and Heckerman, D. (1994). Learning Bayesian networks is NP-hard. Technical report, MSR-TR-94-17, Microsoft Research.

Cimiano, P. and Staab, S. (2005). Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In *Proceedings of the ICML 2005 Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*, pages 168–179, Bonn, Germany.

Cimiano, P. and Völker, J. (2005). Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of RANLPâĂŹ05*, pages 166–172, Borovets, Bulgaria.

Colace, F., De Santo, M., Foggia, P., and Vento, M. (2003). Ontology learning through bayesian networks. In *Proceedings of the 5th International Conference on Enterprise Information Systems*, pages 430–433, Angers, France.

Cooper, G. (1990). Computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405.

Cooper, G. and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

Cooper, G. F. and Yoo, C. (1999). Causal discovery from a mixture of experimental and observational data. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 116–125.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms, Second Edition*. MIT Press, Cambridge, Massachusetts, London, England.

Costa, P. and Laskey, K. (2006). PR-OWL: A Framework for Probabilistic Ontologies. *Frontiers In Artificial Intelligence and Applications*, 150:237–249.

Devitt, A., Danev, B., and Matusikova, K. (2006). Constructing bayesian networks automatically using ontologies. In *Second Workshop on Formal Ontologies Meets Industry*, Trento, Italy.

Ding, Z. and Peng, Y. (2004). A probabilistic extension to ontology language OWL. In *Proceedings of the 37th Hawaii International Conference On System Sciences*, Big Island, HI, USA.

Ding, Z. and Peng, Y. (2005). Modifying bayesian networks by probability constraints. In *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence*, Edinburgh, Scotland.

Essaid, A. and Ben Yaghlane, B. (2009). BeliefOWL: An Evidential Representation in OWL Ontology. In *5th International Workshop on Uncertainty Reasoning for the Semantic Web*, Washington, D.C., USA.

Faatz, A. and Steinmetz, R. (2004). Precision and recall for ontology enrichment. In *Proceedings of ECAI-2004 Workshop on Ontology Learning and Population*, Valencia, Spain.

Fenz, S., Tjoa, M., and Hudec, M. (2009). Ontology-based generation of bayesian networks. In *Proceedings of the Third International Conference on Complex, Intelligent and Software Intensive Systems*, pages 712–717, Fukuoka, Japan.

Fleischman, M. and Hovy, E. (2002). Fine grained classification of named entities. In *Proceedings of ACL-2002*, pages 1–7, Morristown, NJ, USA.

Flouris, G., Plexousakis, D., and Antoniou, G. (2006a). A classification of ontology change. In *The Poster Session of semantic Web Applications and Perspectives (SWAP). 3rd Italian Semantic Web Workshop*, PISA, Italy.

Flouris, G., Plexousakis, D., and Antoniou, G. (2006b). Evolving ontology evolution. In *Proceedings of the 32nd Conference on Current Trends in Theory and Practice of Computer Science*, pages 14–29.

Friedman, N. (1998). The bayesian structural EM algorithm. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 129–138, University of Wisconsin Business School, Madison, Wisconsin, USA. Morgan Kaufmann.

Gómez-Pérez, A., Fernández-López, M., and Corcho, O. (2004). *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-commerce and the Semantic Web*. Springer-Verlag New York, Inc.

Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Gruber, T. (1995). Toward principles for the design of ontologies used for knowledge sharing. *International Journal ol Human-Computer Studies*, 43(5–6):907–928.

Gyftodimos, E. and Flach, P. (2002). Hierarchical bayesian networks: A probabilistic reasoning model for structured domains. In *Proceedings of the ICML-2002 Workshop on Development of Representations*, pages 23–30, University of New South Wales, Sydney, Australia.

Heckerman, D. (1998). A tutorial on learning with Bayesian network. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, pages 301–354, Kluwer Academic Publishers Norwell, MA, USA.

Helsper, E. M. and Van Der Gaag, L. C. (2002). Building bayesian networks through ontologies. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 680–684, Amsterdam.

Henrion, M. (1988). Propagating uncertainty in bayesian networks by probabilistic logic sampling. *Uncertainty in Artificial Intelligence*, 2:149–163.

Keet, C. M. (2010). Ontology engineering with rough concepts and instances. In *17th International Conference on Knowledge Engineering and Knowledge Management by the Masses (EKAW'2010)*, Lisbon, Portugal.

Khattak, A. M., Latif, K., Lee, S. Y., and Lee, Y. K. (2009). Ontology evolution: A survey and future challenges. In *The 2nd International Conference on u-and e- Service, Science and Technology*, Jeju, Koreal.

Khattak, A. M., Pervez, Z., Lee, S. Y., and Lee, Y. K. (2010). After effects of ontology evolution. In *5th International Conference on Future Information Technology*, pages 1–6, Busan, South Korea.

Kim, J. and Pearl, J. (1983). A computational model for combined causal and diagnostic reasoning in inference systems. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 190–193, Karlsruhe, Germany.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models*. MIT Press, Cambridge.

Koller, D. and Pfeffer, A. (1997). Object-oriented bayesian networks. In *Proceedings of the 13th conference on Uncertainty in Artificial Intelligence*, pages 302–313, Providence, Rhode Island, USA. Morgan Kaufmann.

Langset, H. and Bangsø, O. (2001). Parameter learning in object oriented bayesian networks. *Annals of Mathematics and Artificial Intelligence*, 32:221–243.

Langseth, H. and Nielsen, T. D. (2003). Fusion of domain knowledge with data for structural learning in object oriented domains. *Journal of Machine Learning Research*, 4:339–368.

Laskey, K. B. (2008). MEBN: A language for first-order bayesian knowledge bases. *Artificial Intelligence*, 172:140–178.

Leray, P. and Francois, O. (2005). Bayesian network structural learning and incomplete data. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 33–40, Espoo, Finland.

McGarry, K., Garfield, S., Morris, N., and Wermter, S. (2007). Integration of hybrid bio-ontologies using bayesian networks for knowledge discovery. In *Proceedings of the IJCAI-07 Third International Workshop on Neural-Symbolic Learning and Reasoning, NeSy'07*, Hyderabad, India.

McGuinness, D. L. and Harmelen, F. V. (2004). OWL Web Ontology Language - Overview. Retrieved August 20, 2010, from http://www.w3.org/TR/2004/REC-owl-features-20040210/.

Meganck, S., Leray, P., and Manderick, B. (2006a). Learning causal bayesian networks from observations and experiments: a decision theoretic approach. In *Proceedings of the Third International Conference MDAI, Lecture Notes in Artificial Intelligence*, pages 58–69.

Meganck, S., Leray, P., and Manderick, B. (2008). Uncado: Unsure causal discovery. In *Proceedings of 4èmes journées francophones de réseaux Bayésiens JFRB*, pages 5–16.

Meganck, S., Maes, S., Leray, P., and Manderick, B. (2006b). Learning semi-markovian causal models using experiments. In *Proceedings of The third European Workshop on Probabilistic Graphical Models*, pages 195–206.

Mori, J., Matsuo, Y., and Ishizuka, M. (2006). Extracting relations in social networks from the web using similarity between collective contexts. In *Proceedings of the 5th International Semantic Web Conference*, pages 487–500, Athens, GA, USA.

Murphy, K. P. (2001). Active learning of causal Bayes net structure. Technical report, Departement of Computer Science, UC Berkeley.

Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, Computer Science Division.

Naïm, P., Wuillemin, P.-H., Leray, P., Pourret, O., and Becker, A. (2004). *Réseaux bayésiens*. Eyrolles, Paris.

Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the AAAI National Conference on AI*, pages 133–136, Pittsburgh, PA.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, San Franciscos.

Pearl, J. (2000). *Causality: Models, reasoning and inference*. MIT Press, Cambridge.

Pfeffer, A. J. (2000). *Probabilistic Reasoning for Complex Systems*. PhD thesis, Stanford University.

Renooij, S. (2001). Probability elicitation for belief networks: Issues to consider. *Knowledge Engineering Review*, 16(3):255–269.

Ribino, P., Oliveri, A., Lo Re, G., and Gaglio, S. (2009). A Knowledge Management System Using Bayesian Networks. In *Proceedings of the 9th International Conference of the Italian Association for Artificial Intelligence*, pages 446–455, Berlin, Heidelberg. Springer-Verlag.

Ruiz-Casado, M., Alfonseca, E., and Castells, P. (2005). Using context-window overlapping in synonym discovery and ontology extension. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, Borovets, Bulgaria.

Schwarz, G. (1978). Estimating the dimensions of a model. *Annals of Statistics*, 6:461–464.

Spirtes, P. (2010). Introduction to causal inference. *Journal of Machine Learning Research*, 11:1643–1662.

Spirtes, P., Glymour, C., and Scheines, R. (2001). *Causation, Prediction and Search*. MIT Press.

Stoilos, G., Stamou, G., Tzouvaras, V., Pan, J., and Horrocks, I. (2005). Fuzzy OWL: Uncertainty and the Semantic Web. In *International Workshop of OWL: Experiences and Directions*, Galway.

Stojanovic, L., Maedche, A., Stojanovic, N., and Motik, B. (2002). User-driven ontology evolution management. In *The 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW-2002)*, pages 285–300, Siguenza, Spain.

Tanev, H. T. and Magnini, B. (2006). Weakly supervised approaches for ontology population. In *Proceedings of EACL-2006*, Trento, Italy.

Tong, S. and Koller, D. (2002). Active learning for structure in bayesian networks. In *Proceedings of the National Conference on Artificial Intelligence*, pages 567–573.

Torti, L., Wuillemin, P.-H., and Gonzales, C. (2010). Reinforcing the object-oriented aspect of probabilistic relational models. In *Proceedings of the 5th Probabilistic Graphical Models*, pages 273–280.

Verma, T. and Pearl, J. (1990). Equivalence and synthesis of causal models. In *Proceedings of the 6th Conference on Uncertainty and Artificial Intelligence*, San Francisco, Morgan Kaufmann.

Xiang, Y., Poole, D., and Beddoes, M. P. (1993). Multiply sectioned bayesian networks and junction forests for large knowledge-based systems. *Computational Intelligence*, 9(2):171–220.

Xuan, D. N., Bellatreche, L., and Pierra, G. (2006). A versioning management model for ontology-based data warehouses. In *The 8th International Conference on Data Warehousing and Knowledge Discovery*, pages 195–206, Krakow, Poland.

Yang, Y. and Calmet, J. (2005). Ontobayes: An ontology-driven uncertainty model. In *International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, Vienna, Austria.

Zheng, H., Kang, B.-Y., and Kim, H.-G. (2008). An ontology-based bayesian network approach for representing uncertainty in clinical practice guidelines. In *Uncertainty Reasoning for the Semantic Web I: ISWC International Workshops, URSW 2005-2007, Revised Selected and Invited Papers*, pages 161–173.