Ontology-based generation of Object Oriented Bayesian Networks

Mouna Ben Ishak
LARODEC Laboratory
ISG, University of Tunis
Tunisia, 2000
mouna.benishak@gmail.com

Philippe Leray
Knowledge and Decision Team
LINA Laboratory UMR 6241,
Polytech'Nantes, France
philippe.leray@univ-nantes.fr

Nahla Ben Amor
LARODEC Laboratory
ISG, University of Tunis
Tunisia, 2000
nahla.benamor@gmx.fr

Abstract

Probabilistic Graphical Models (PGMs) are powerful tools for representing and reasoning under uncertainty. Although useful in several domains, PGMs 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 applications. Ontologies, from their side, provide a body of structured knowledge characterized by its semantic richness. This paper proposes to harness ontologies representation capabilities in order to enrich the process of PGMs building. We are in particular interested in object oriented Bayesian networks (OOBNs) which are an extension of standard Bayesian networks (BNs) using the object paradigm. We show how the semantical richness of ontologies might be a potential solution to address the challenging field of structural learning of OOBNs while minimizing experts involvement which is not always obvious to obtain. More precisely, we propose to set up a set of mapping rules allowing us to generate a prior OOBN structure by morphing an ontology related to the problem under study to be used as a starting point to the global OOBN building algorithm.

1 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.

Probabilistic graphical models (PGMs) provide an efficient framework for knowledge representation and reasoning under uncertainty. Ontologies allow logical reasoning about concepts linked by semantic relations within a knowledge domain. Even though they represent two different paradigms, PGMs and ontologies share several similarities which has led to some research directions aiming to combine them. 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 enrich ontology reasoning with probabilistic queries. Few works intend to construct PGMs using ontologies. In this area, Bayesian networks (BNs) (Pearl, 1988) 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. To overcome this weakness, we propose to explore other PGMs, significantly more expressive than standard BNs, in order to address an extended range of ontologies.

We are in particular interested in *object oriented Bayesian networks* (Bangsø and Wuillemin, 2000) (OOBN), which are an extension of standard BNs. In fact, OOBNs share several similarities with ontologies and they are suitable to represent hierarchical systems as they introduce several aspects of object oriented modeling, such as inheritance. Our idea is to benefit from ontologies in order to address the challenging problem of OOBN structure learning known to be an NP-hard process. To this end, we first establish the correspondence between OOBNs and ontologies. Then, we describe how to generate a prior OOBN structure by morphing an ontology related to the problem under study and then to use it as a starting point to the global building OOBN algorithm. This latter will take advantages from both semantical data, de-

rived from ontology which will ensure its good start-up and observational data.

The remainder of this paper is organized as follows: In sections 2 and 3 we provide a brief representation of our working tools. In section 4, we show how to benefit from knowledge provided by an ontology to define the structure of an OOBN. In section 5 we represent a survey on existing approaches trying to find a combination between PGMs and ontologies. The final section summarizes conclusions reached in this work and outlines directions for future research.

2 Object Oriented Bayesian Networks

Probabilistic graphical models (PGMs) provide an efficient framework for knowledge representation and reasoning under uncertainty. In the literature, we distinguish a panoply of PGMs sharing two common components: a graphical one (i.e. a set of nodes and links) and a numerical one allowing the quantification of different links defined in the graphical component via probability distributions. Among the most used PGMs we can mention Bayesian networks (BNs) (Pearl, 1988) which have been largely developed and used in several real world applications. 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), (Koller and Pfeffer, 1997) which introduce the object oriented paradigm into the framework of BNs. 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). Thus an 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, which are not instantiations of classes, 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.

When some classes in the OOBN are similar (i.e. share some nodes and potentials), their specification can be simplified by creating a class hierarchy among them. 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 $\mathcal{H}_T \subseteq \mathcal{H}_S$.

Example 1. *Figure 1 represents the insurance network adapted to the OOBN framework (Langseth and Nielsen, 2003). This network contains six classes (Insurance, Theft, Accident, Car, CarOwner and Driver). In this figure only the interfaces of the encapsulated instantiations are shown, dashed ellipses represent input nodes, while shaded ellipses represent output nodes. For instance, the class CarOwner describes properties of a car owner. It has no input nodes, the nodes Age, SocioEcon, HomeBase, AntiTheft, VehicleYear and MakeModel operate as output nodes of this class. Moreover, Driven characteristics are a part of the notion of a car owner. Thus, an instantiation of the class Driver is then encapsulated in the class CarOwner. Note that the output node DrivQuality of the class Driver is used as output reference node of the class CarOwner as it is referenced in the Accident class.*

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.

As all PGMs, OOBNs have two fundamental cornerstones: construction and reasoning. The construction of an OOBN concerns both learning the graph structure and parameters estimation. Few works have been proposed in the literature to learn the structure (Bangsø et al., 2001), (Langseth and Nielsen, 2003) and the parameters (Langseth and Bangsø, 2003) of such a model from data. Given an OOBN, reasoning

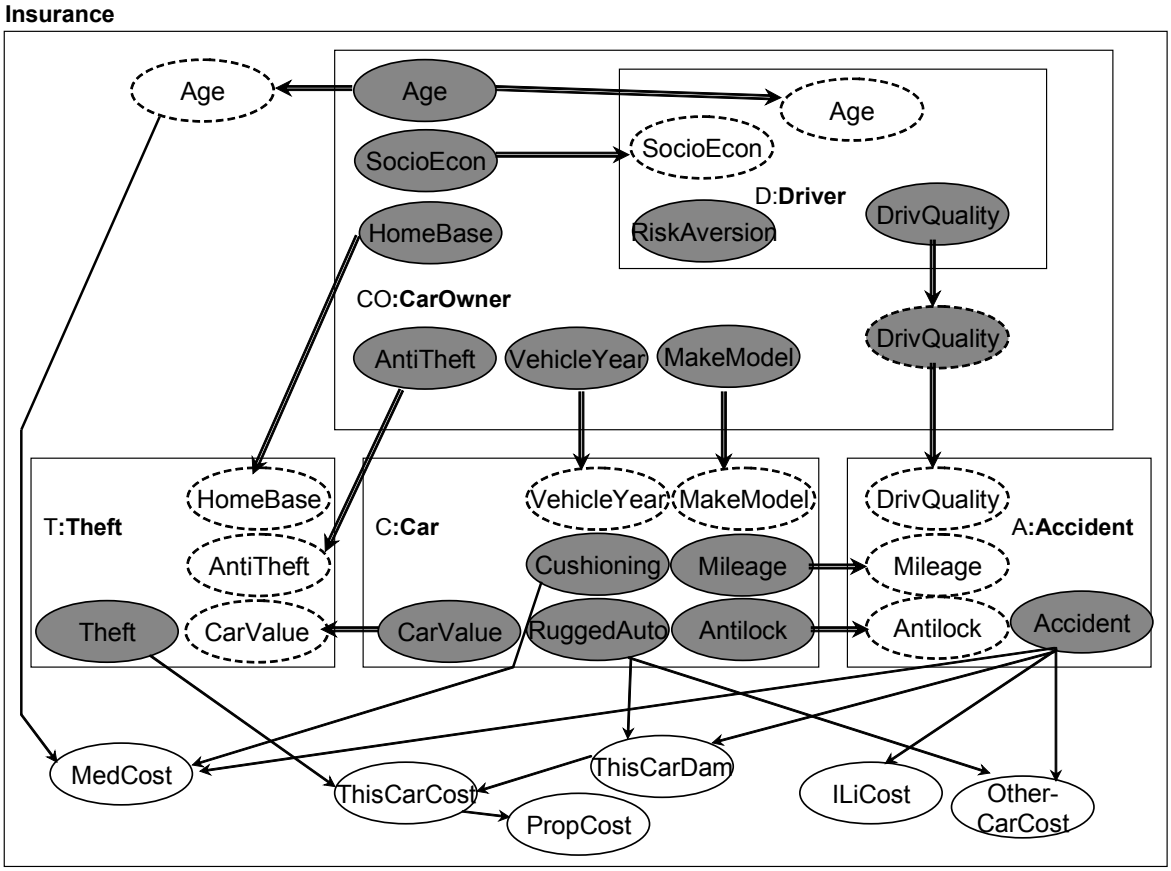


Figure 1: The insurance network represented using the OOBN framework

stands for probabilistic inference and this requires to translate the OOBN into a BN or a multiply sectioned Bayesian network (MSBN) (Bangsø and Wuillemin, 2000).

In this paper we are interested in the learning process. The standard approach proposed by (Langseth and Bangsø, 2003) is the OO-SEM algorithm. This algorithm is based on an **Object Oriented assumption** which states that *all instances of a class are assumed to be identical w.r.t. both parameters and structure*. This algorithm is based on a prior expert knowledge about 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 the OOBN structure that fits best to the data. Learning in object oriented domains allows to reduce the search space, however it remains an NP-hard problem. In fact, 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. However, human expertise, required to initiate the learning process, is not always obvious to obtain. To overcome this limitation we propose to use ontologies richness. Before introducing our method, we give basic notions on ontologies.

3 Ontologies

Over the last few years, there has been an increasing interest in the application of ontologies in various domains (e.g., linguistics, semantic web, bioinformatics). They represent not only a fixed structure but also the basis for deductive reasoning. For the AI community, an ontology is *an explicit specification of a conceptualization* (Gruber, 1995). That is, an ontology is a description of a set of representational primitives with which to model an abstract model of a knowledge domain. Formally, we define an ontology $\mathcal{O} = \langle \mathcal{C}p, \mathcal{R}, \mathcal{I}, \mathcal{A} \rangle$ as follows:

- $\mathcal{C}p = \{cp_1, \dots cp_n\}$ is the set of n concepts (classes) such that each cp_i has a set of k properties (attributes) $\mathcal{P}_i = \{p_1, \dots p_k\}$.
- $\mathcal{R}$ is the set of binary relations between elements of $\mathcal{C}p$ which consists of two subsets:

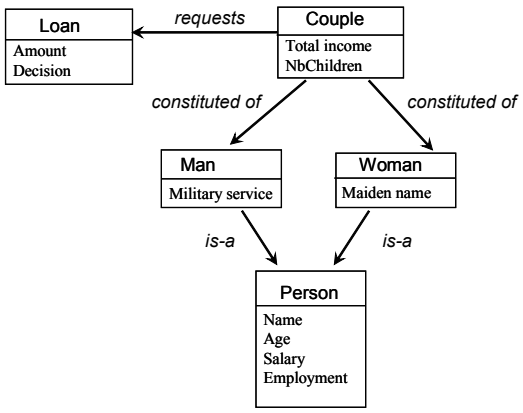


Figure 2: The joint credit ontology

- $\mathcal{H}_{\mathcal{R}}$ 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 $\mathcal{I}$.

Axioms are of the form $A \equiv B$ (A and B are equivalent), $R1 \subseteq R2$ ($R1$ is a subproperty of $R2$), $R1(x, y)$ (x is related to y by the relation $R1$), $A(x)$ (x is of type A), etc. Where A and B are concepts, $R1$ and $R2$ are relations and x and y are instances.

Example 2. *Figure 2 is an example of a joint credit ontology.*
$\mathcal{C}p = \{Loan, Couple, Man, Woman, Person\}$.
For instance $\mathcal{P}_{Loan} = \{Amount, Decision\}$.
$\mathcal{S}_{\mathcal{R}}$ = {*requests*$(Couple \times Loan)$, *constituted of*$(Couple \times Man)$, *constituted of*$(Couple \times Woman)$}. *Is-a relations represent* $\mathcal{H}_{\mathcal{R}}$ *and are equivalent to subsumption axioms, e.g., Man*$\subseteq$ *Person. For instance, we can have an instance p of the concept Man* $p\{Paul, 30, 2000\$, teacher, T\}$, *p.Name = Paul, p.Age = 30, p.Salary = 2000\$, p.Employment = teacher and p.Military service = T.*

4 A new approach for OOBNs building based on ontologies

Clearly PGMs and ontologies share several similarities even they are derived from different frameworks. Thus our idea is to use the ontological knowledge in the OOBN learning process by morphing the ontology in hand into the a prior OOBN structure. To this end, we first define the common points and similarities between these two paradigms, then we describe the main steps of our proposal.

4.1 OOBNs vs ontologies

In this part, we highlight the common points and similarities between ontologies and object oriented Bayesian networks. The main components of an ontology (i.e., concepts and relations) may be viewed as a start-up to define the main components of an OOBN (i.e., classes and relations among them).

- **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 the concept properties does not cover the three sets of nodes of 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.

- **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.

- **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 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 a 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_i is considered as the cause of cp_j and this latter is the effect.

In fact, the ontology conceptual graph is simply the result of the ontology components definition. Thus, we require that semantic relations definition to be, from the beginning, done following a causal reasoning that is considered as an intuitive reflexion of the ontologist. Then, if we require to have all semantic relations to be anti-causal, we just have to reverse their definitions (i.e., define the domain as range and vice versa).

4.2 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. The idea over the Depth-First Search algorithm is to traverse a graph by exploring all the vertices reachable from a source vertex: If all its neighbors have already been visited (in general, color markers are used to keep track), or there are no ones, then the algorithm backtracks to the last vertex that had unvisited neighbors. Once all reachable vertices have been visited, the algorithm selects one of the remaining unvisited vertices and continues the traversal. It finishes when all vertices have been visited. The DFS traversal allows us to classify edges into four classes:

- Tree edges: are edges in the DFS search tree.
- Back edges: join a vertex to an ancestor already visited.
- Forward edges: are non-tree edges connecting a vertex to a descendant in a DFS search tree.
- Cross edges: all other edges.

We use these classes of edges to determine actions to do on each encountered concept. Tree edges allow to define actions on concepts encountered for the first time, while 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. According to their definition, 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 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:

i. **Initialization step:** All concepts are undiscovered, we generate the OOBN class and a class to

Algorithm 1: Generate_OOBN

Input: An ontology $\mathcal{O}$
$\mathcal{O}$ is of an anti-causal orientation.

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

begin
  CREATE_OOBN_GLOBAL ;
  for *each concept* $cp \in \mathcal{C}p$ **do**
    RECORD_PREDECESSOR[cp]=NULL;
    CREATE_CLASS(cp)
  for *each concept* $cp \in \mathcal{C}p$ **do**
    if *color[cp]= white* **then**
      Handling_Process($\mathcal{O}$, cp)

each concept:

CREATE_OOBN_GLOBAL : creates the OOBN class.

CREATE_CLASS(Concept cp): transforms a concept cp to a class c_{cp}.

ii. **Discovery step:** The classes of edges are 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 or from different paths. Let cp_i be a concept having two parents $cp1_i$ and $cp2_i$ coming from two different branches. Then, c_{cp_i} would to be instantiated within the specification of only one of them. However, c_{cp_i} has its output nodes

Algorithm 2: Handling_Process

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

We use color markers to keep track of which vertices have been discovered: white marks vertices that have yet to be discovered, gray marks a vertex that is discovered but still has vertices adjacent to it that are undiscovered and black marks discovered vertex that is not adjacent to any white vertices.

begin
  color[S]:= gray;
  for *each property p of S* **do**
    ADD_OUTPUT_NODE(p, c_S)
  for *each* $V \in adjacent[S]$ **do**
    if *color[V]=white* **then**
      RECORD_PREDECESSOR[V]=S;
      Handling_Process($\mathcal{O}$, 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* $n \in GET\_OUTPUT(c_V)$ **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* $n \in GET\_OUTPUT(c_V)$ **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)

to be referenced by output reference nodes of the class containing it until reaching its second parent (see figure 3).

ADD_CONSTRUCT_LINK(Class $c1$, Class $c2$): a construct link appears between instantiations of superclasses and instantiations of their subclasses (see figure 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.

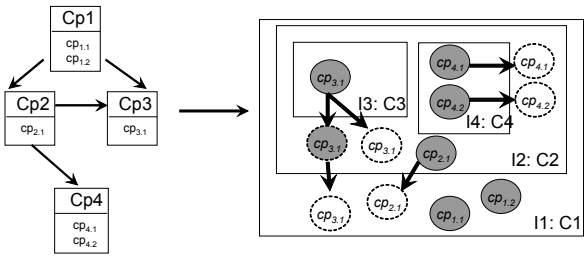


Figure 3: An example of more than one parent

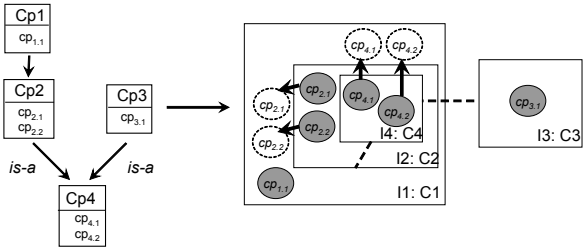


Figure 4: An example of inheritance relation

iii. **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 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 1 and 2. The *Handling_Process* function (see algorithm 2) provides actions to do at each vertex.

Example 3. *We assume that all random variables are modeled in the corresponding ontology 5(b) as concepts properties and that all semantic relations present in the ontology are of an anti-causal orientation.*

We will follow the steps of the Generate_OOBN algorithm (see algorithm 1)to generate our prior OOBN structure.

First of all, we start by generating the Global OOBN class Prior_OOBN. Then we create a class to each concept of the ontology, that is, we create 11 classes $c_{cp_i}, i = \{1, \ldots 11\}$ *which are initially empty. their sets of nodes will be discovered during the generation process.*

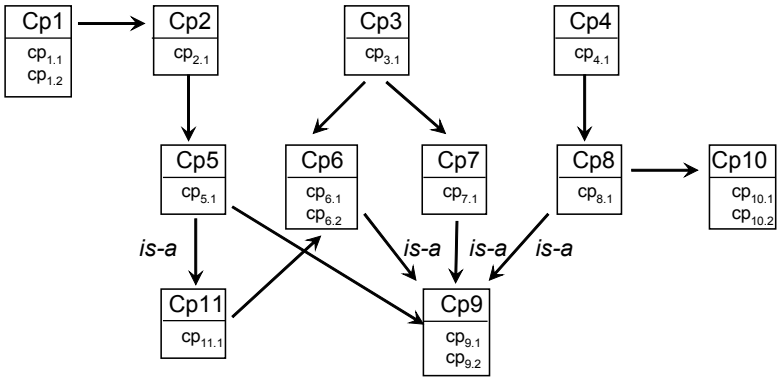
Initially, all concepts are white. cp_1 *is the source concept, it is grayed and all its properties are declared as output nodes of the class representing it in the prior OOBN. Then, each concept adjacent to* cp_1 *is recursively visited if it is white and its properties are treated in the same way.* cp_1 *has* cp_2 *as adjacent concept, it is painted gray and it has* cp_5 *as adjacent concept, so on until reaching* cp_9*.* cp_9 *is grayed and all its properties are declared as output nodes of the class representing it in the prior OOBN (*c_{cp_9}*). As it has no adjacent, it is instantiated within its ancestor* c_{cp_6}*. As* cp_6 *and* cp_9 *are related by an inheritance relation then, we add a construction link between* c_{cp_6} *and* c_{cp_9} *and all properties of the super-class* c_{cp_9} *are considered as properties of its subclass* c_{cp_6}*. The concept* cp_9 *is finished and blackened. We backtrack to the* cp_6 *concept, it is gray and it has finished his adjacent concepts so, it is instantiated within its ancestor* $c_{cp_{11}}$*. As* cp_{11} *and* cp_6 *are related by a semantic relation then, all its output nodes are considered as input nodes of the* $c_{cp_{11}}$ *class linked by reference links.* cp_6 *is blackened and we go back to* cp_{11} *it is gray and it has finished his adjacent concepts so, it is instantiated within its ancestor* c_{cp_5}*. As* $c_{cp_{11}}$ *and* c_{cp_5} *are related by an inheritance relation then, we add a construction link between them.* cp_{11} *is blackened and we go back to the second adjacent of the* cp_5 *concept.* cp_5 *is gray and* cp_9 *is black, so (*cp_5*,*cp_9*) is a cross/forward edge, means that* cp_9 *has already been instantiated so, we add output reference nodes from* c_{cp_6} *until reaching the* c_{cp_5} *class.* cp_5 *is gray and it has finished his adjacent concepts so, it is instantiated within its ancestor and so on until backtracking to the concept* c_{cp_1}*, it is gray and it has not ancestors, so it is blackened and instantiated within the specification of the Prior_OOBN class. The first DFS tree is finished, so we choose an undiscovered node from the remainder nodes and we apply the algorithm until discovering all the concepts. The result of this process is shown in figure 5.(b)) shows the final result.*

5 Related work

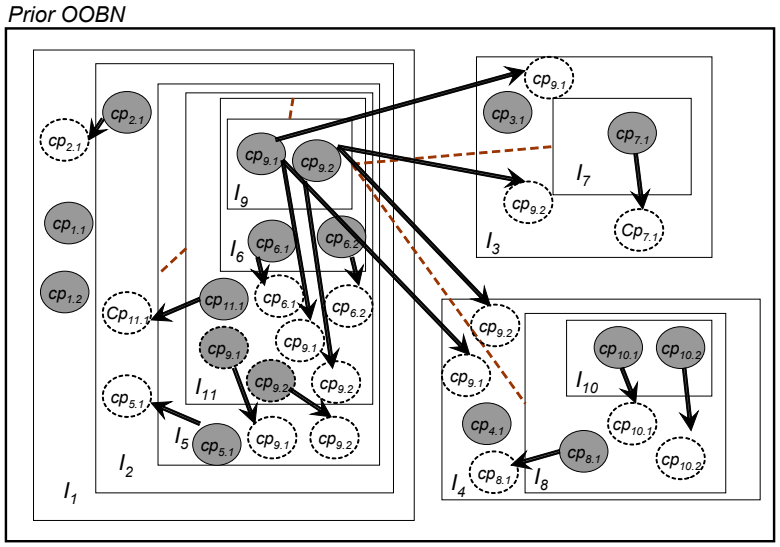
In recent years, a panoply of works have been proposed in order to combine PGMs and ontologies so that one can enrich the other. We can outline two main directions for these proposed approaches. The first aims to enhance ontologies capabilities to support probabilistic inference. While the second aims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

Ontologies provide a support for logical reasoning, but they do not support uncertainty. Hence, several extensions have been proposed to overcome this limitation. One line of research aims to extend existing ontology languages, such as OWL[1], to be able to catch uncertainty in the knowledge domain. Proposed methods, such as BayesOWL (Ding and Peng, 2004), OntoBayes (Yang and Calmet, 2005) use additional markups to represent probabilistic information attached to indi-

[1]ttp://www.w3.org/TR/2004/REC-owl-features-20040210/



(a) The ontology to morph



(b) The resulting OOBN

Figure 5: An example of ontology morphing to a prior OOBN structure

vidual concepts and properties in OWL ontologies. The result is then a probabilistic annotated ontology that can be translated into a BN to perform probabilistic inference. Other works define transition actions in order to generate a PGM given an ontology with the intention of extending ontology querying to handle uncertainty while keeping the ontology formalism intact (Bellandi and Turini, 2009).

On the other hand, some solutions proposed the use of ontologies to help PGMs construction. Some of them are designed for specific applications (Helsper and Van der Gaag,2002), (Zheng et al., 2008), while some others give various solutions to handle this issue. We can mention the semi-automatic approach provided in (Fenz et al., 2009) to create BNs and the SemCaDo (Semantical Causal DiscOvery) algorithm (Ben Messaoud et al., 2009) (Ben Messaoud et al., 2011) which ensure the integration of ontological knowledge, more precisely, subsumption relationships, to learn the structure of causal Bayesian networks (i.e. BNs with causal relations) (Pearl, 2000) and improve the causal discovery.

However, all these solutions are limited to a restrained range of PGMs, usually BNs. So, they neglect some ontology important aspects such as representing concepts having more than one property, non taxonomic relations, etc. In our approach we used OOBNs which are much richer graphical model than standard BNs. They allowed us to address an extended range of ontologies, we focused on concepts, their properties, hierarchical as well as semantic relations and we showed how these elements would be useful to automatically generate a prior OOBN. Our proposal concerns exclusively the OOBN structure definition through the use of ontologies.

6 Conclusion and future work

The crossing-over of PGMs and ontologies can allow us to improve relevant tasks related to each of them. In this paper, we showed how we take advantage of the semantic richness provided by ontologies to generate a prior 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, which brings us to say that this work is an initiative aiming to set up new bridges between these two paradigms.

The final structure resulting from the learning process may also be useful to make the initial ontology evolve, and this is by trying to find how the new relations discovered by the learning process can affect the (semi) automatic ontology enrichment process. Thus, as an ongoing work, we aim to analyze the elements that are common to both tasks and provide a two-way approach that uses ontology power in representing knowledge to help the hard process of OOBN structure learning by proposing new metrics, based on ontological knowledge, allowing to assess better the choice of the best structure. Then, uses novel relations discovered by the learning process in order to improve the hard activity of ontology enrichment.

References

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

O. Bangsø and P-H. Wuillemin (2000). Object Oriented Bayesian networks: a framework for top-down specification of large Bayesian networks with repetitive structures. Aalborg University, Denmark.

A. Bellandi and F. Turini (2009). Extending Ontology Queries with Bayesian Network Reasoning. *Proceedings of the 13th International Conference on Intelligent Engineering Systems*, 165–170. Barbados.

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

M. Ben Messaoud, Ph. Leray 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*, 168–179. Verona, Italy.

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

S. Fenz, M. Tjoa and M. Hudec (2009). Ontology-Based Generation of Bayesian Networks. *Proceedings of the Third International Conference on Complex, Intelligent and Software Intensive Systems*, 712–717. Fukuoka, Japan.

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

T. Gruber (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies* **43** (5–6): 907–928.

E. M. Helsper and L. C. van der Gaag (2002). Building bayesian networks through ontologies. *Proceedings of the 15th European Conference on Artificial Intelligence*, 680-684. Amsterdam.

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

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

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

H. Langseth and O. Bangsø (2001). Parameter learning in object oriented Bayesian networks. *Annals of Mathematics and Artificial Intelligencen*, **32**:221-243.

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

Y. Yang and J. Calmet (2005). OntoBayes: An Ontology-Driven Uncertainty Model. *International Conference on Intelligent Agents, Web Technologies and Internet Commerce*. Vienna, Austria.

H-t. Zheng, Bo-Y. Kang and H-G. Kim (2008). An Ontology-Based Bayesian Network Approach for Representing Uncertainty in Clinical Practice Guidelines. *Uncertainty Reasoning for the Semantic Web I: ISWC International Workshops, URSW 2005-2007, Revised Selected and Invited Papers*, 161–173.