République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

Université Saad Dahlab Blida

N° D'ordre : .....

Faculté des sciences
Département d'informatique
Mémoire Présenté par :

BENSALEM Yamina

MANSOURI Ahlem

En vue d'obtenir le diplôme de master

Domaine : mathématique et informatique

Filière : Informatique
Spécialité : Informatique
Option : Ingénierie de logiciel

Thème : Modélisation de l'incertitude dans le web sémantique en utilisant les réseaux bayésiens

Soutenu le : 25/06/2016

Mme : ZAHRA	Présidente
Mme : HAJ HENNI	Examinatrice
Mme : FAREH	Promotrice

Promotion
2015 / 2016

Dédicaces

A :

Nos chers parents

Nos frères et sœurs

Toute Notre famille

Tous nos amis

On dédie ce mémoire

BENSALEM YAMINA

MANSOURI AHLEM

Remerciement

On remercie en premier lieu le tout puissant de nous avoir donné la chance, le courage et les moyens pour l'accomplissement de ce projet.

On remercie tout particulièrement Mm FAREH.M enseignante à l'université USDB, pour nous avoir aidés et épaulé durant toute cette année, de nous avoir guidé et soutenu dans la réalisation du projet, merci.

On remercie le doctorant Mr RIALI Ishak pour nous avoir aidé et suivi durent la réalisation de ce projet, merci.

On remercie tous les membres du jury d'avoir accepté d'examiner notre travail.

Au terme de ce mémoire, On tient à remercie toute personne ayant contribué à l'accomplissement de ce travail de prêt ou de loin et qui nous ont été bénéfique durant notre parcours pour la réalisation de ce travail.

Finalement, On remercie l'ensemble des enseignants du département d'informatique pour les efforts qu'ils fournissent pour notre réussite dans nos études.

BENSALEM YAMINA

MANSOURI AHLEM

Résumé :

Les ontologies sont au cœur du web sémantique, elles permettent de représenter les connaissances du web. Une ontologie est une représentation qui regroupe un ensemble de concepts et relations décrivant un domaine particulier. Un de principaux défauts de l'ontologie est leur incapacité de représenter et raisonner sur l'incertitude, d'où la modélisation des connaissances du web sémantique.

Nous proposons dans ce mémoire une méthode de modélisation de l'incertitude plus particulièrement l'incomplétude dans les ontologies, en utilisant un langage probabiliste à base des réseaux bayésiens, et en appliquant des algorithmes dédiés à l'apprentissage des paramètres et des structures de ces réseaux bayésiens pour remplir les tables de PR-OWL.

Mots clé: **OWL, PR-OWL, MEBN, ontologies, réseaux bayésiens, MTheory, MFrag, nœuds.**

Abstract:

Ontologies are the heart of the Semantic Web; they can represent knowledge of the web. Ontology is a representation that includes a set of concepts and relationships describing a particular area. A main defect of ontology is their inability to represent and reason about uncertainty, hence modeling semantic web knowledge.

We propose in this paper a method for modeling uncertainty in ontologies using a probabilistic language based on Bayesian networks, and applying algorithms dedicated to learn parameters and structures of these Bayesian networks in order to fill the PR-OWL tables.

ملخص:

الأونتولوجيا هي قلب الويب الدلالي. حيث أنها يمكن أن تمثل المعرفة على شبكة الإنترنت. الأنطولوجيا هو تمثيل يتضمن مجموعة من المفاهيم والعلاقات واصفا مجال معين. المشكل الرئيسي للأنطولوجيا هو عدم قدرتهم على تمثيل الغموض و عدم التأكد، ومنه عدم قدرتهم على نمذجة المعرفة على شبكة الإنترنت الدلالي.

نقترح في هذه المذكرة طريقة لنمذجة عدم التأكد في الأنطولوجيات باستخدام لغة الاحتمالات القائمة على شبكات النظرية الافتراضية، وتطبيق خوارزميات مخصصة لمعلمات وهياكل هذه الشبكات النظرية الافتراضية لملء جداول براول.

Sommaire :

Chapitre 3: Langages probabilistes à base des réseaux bayésiens

Chapitre 4: conception du système

Chapitre 5: Implémentation du système

Liste des figures :

Liste des tableaux :

LISTE DES ABREVIATIONS:

OWL: Ontology Web Language.

PR-OWL: PRobabilistic Ontology Web Language.

W3C: World WideWeb Consortium.

URI: Uniform Ressource Identifier.

XML: eXtensible Markup Language.

RDF: Resource Description Framework.

RDFS: Resource Description Framework Schema.

SBC: Systèmes à Base de Connaissances.

KIF: Knowledge Interchange Format.

OIL: Ontology Interchange Language and Ontology Inference Layer.
DAML+OIL DARPA: Agent Markup Language.

OWL DL: Ontology Web Language Description Logic.

RB: Réseau Bayésien.

DAG: directed acyclic graph.

EM: Expectation-Maximisation.

MEBN : Multy Entity Bayesian Network.

FOL: First Order Logic.

Introduction générale:

Le domaine de la sémantique a été extrêmement actif au cours des dernières années. Dans un tel contexte, les technologies du Web sémantique, tels que les ontologies ont émergé comme un Framework d'intégration pour enregistrer, distinguer, exploiter et intégrer les ressources web.

La capacité à soutenir le raisonnement déductif fait les ontologies un outil prometteur pour la gestion des connaissances, les langages d'ontologie traditionnels, tels que le Web Ontology Language – OWL permettent aux ordinateurs de déduire efficacement des significations multiples pour un mot, mais il n'a pas la capacité d'évaluer finement leurs plausibilités respective dans les contextes. Cette limitation provient directement du fait que le raisonnement déductif dans ontologies traditionnelles repose sur différentes saveurs de la logique classique pour soutenir le raisonnement automatisé.

Bien qu'acceptable dans des domaines bien définis avec des sources d'information complètes et cohérentes, ceci est une sérieuse limitation lorsqu'on fait des traitements dans des environnements mondiaux ouverts tels que le Web sémantique, dans lequel l'information provient de diverses sources et est souvent incomplète, Inexactes ou peu fiables.

Ces circonstances, notamment, peuvent conduire à une mauvaise réutilisation et intégration des connaissances, et renforcer le besoin des moyens de représenter des informations incertaines d'une manière raisonnée [1]. Les représentations basées sur le langage naturel sont intrinsèquement incertaines.

Problématique :

Les ontologies traditionnelles (formelles) sont vraiment puissantes, mais leurs base de la logique classique ne fournit aucun support intégré pour l'incertitude, ceci est un inconvénient majeur pour les applications du Web sémantique, qui sont souvent nécessaires pour effectuer le raisonnement automatisé dans des environnements incertains et complexes.

L'utilisation des modèles probabilistes graphiques est un moyen attrayant de permettre des systèmes d'information à exploiter l'incertain et les informations incomplètes de façon cohérente, les réseaux bayésiens (RB) , plus précisément les réseaux bayésiens multi entités MEBN sont un moyen graphique, flexible pour exprimer des distributions de probabilités conjointes sur des hypothèses interdépendantes.

Etant donné une ontologie qui peut contenir des connaissances incertaines qui due à l'incomplétude, nous voudrons créer une autre ontologie probabiliste, ainsi, nous voudrons automatiser la construction de la structure de cette ontologie probabiliste ainsi que ses tables de probabilité

Objectif :

Les réseaux bayésiens sont un outil souple et performant de gestion de l'incertitude. Un des principaux avantages de RB est leur capacité de gérer l'incertitude: cela permet de représenter les informations incertaines avec une manière très lisible et clair et offre des techniques puissantes d'inférences bayésiennes qui consiste à propager une ou plusieurs informations (valeurs probabilistes) dans le réseau pour en déduire comment ceci intervient sur les probabilités d'autres variables.

L'objectif de ce projet est de formaliser la connaissance incertaine et de l'intégrer dans l'ontologie, à fin de la rendre compréhensible par les utilisateurs, en utilisant les réseaux bayésiens, pour permettre de raisonner sur les connaissances incertaines, et de créer un outil qui génère une ontologie probabiliste PR-OWL d'un domaine précis et qui résout le problème des déclarations manuelles des probabilités en utilisant des algorithmes spécifiques pour l'apprentissage des données incomplètes, et qui optimise la structure du réseau bayésien.

Afin d'atteindre le but de notre travail, le mémoire sera présenter en deux parties :

La première partie :

Contient deux chapitres, elle présente le contexte du travail, elle a pour but de présenter les ontologies, et les réseaux bayésiens ainsi que la définition et la syntaxe du langage PR-OWL.

Chapitre 1- Les Ontologies : Ce chapitre propose une présentation du web sémantique et une généralité sur les ontologies. Nous présentons le web sémantique et l'incertain dans le web sémantique, les ontologies avec leurs caractéristiques, leurs constructions, ainsi que les domaines d'application de ces ontologies. Nous allons décrire quelques langages de représentation des ontologies utilisées dans le Web sémantique, et nous allons introduire la notion des ontologies probabilistes

Chapitre 2- Les réseaux bayésiens : Ce chapitre a pour objectif de donner une vision sur les réseaux bayésiens et sur un langage probabiliste à base des réseaux bayésiens. Tout au long de ce chapitre, nous éclaircissons la notion des réseaux bayésiens en présentant leurs différentes méthodes d'apprentissage et d'inférence. Nous présentons MEBN qui fait l'objet de notre application.

Chapitre 3- langage probabiliste à base des réseaux bayésiens PR-OWL : dans ce chapitre, nous présentons les définitions d'un langage probabiliste à base des réseaux bayésiens qui est le PR-OWL.

La deuxième partie :

Après les généralités présentées dans la première partie. Cette partie est consacrée pour la conception et le développement du système.

Cette partie met l'accent sur la conception de notre système de la modélisation de l'incertitude dans une ontologie OWL.

Chapitre 3- La conception du système : Ce chapitre contient les étapes principales pour la construction d'une ontologie PR-OWL, ainsi que les étapes de l'intégration de l'algorithme K2 pour l'apprentissage d'une structure optimisée, et de l'algorithme EM pour l'apprentissage des données manquantes, cela se fait à partir d'un schéma qui représente le système globale de la modélisation et qui sera détaillé par la suite dans ce chapitre. Ce chapitre constitue un point de départ à l'implémentation.

Chapitre 4: Implémentation et test du système: Ce chapitre est le résultat de la conception pour réaliser le système de la modélisation de l'ontologie probabiliste PR-OWL, il s'agit de transformer les éléments décrits lors de la conception en éléments du langage cible. A la fin de ce chapitre on va effectuer un test sur notre application.

La conclusion de ce mémoire synthétise les principales contributions de notre travail.

Chapitre 1 : les ontologies

1. Introduction :

Les ontologies sont la technologie de base du web sémantique et pour le management des connaissances formalisées décrivant les ressources du web. La principale raison à cela est qu'elles fournissent la sémantique des données d'un document web, ainsi que les sources d'informations qui peuvent être transmises entre différents agents logiciels. Les métadonnées et annotations sémantiques décrivent les ressources en utilisant la sémantique définie dans l'ontologie. De ce fait, les ressources annotées via les métadonnées permettront de faciliter la recherche, l'extraction, l'interprétation et le traitement des informations du web de façon efficace.

2. Le web sémantique :

2.1 Définition :

Le web sémantique constitue un environnement dans lequel les humains et les machines vont communiquer selon une base sémantique. La première définition qui ressort lorsque l'on parle de web sémantique est celle d'un web compréhensible par les machines. Il s'agit donc d'un concept à la fois large et complexe définissable de la façon suivante : ensemble de technologies visant à rendre le contenu des ressources du web accessible et utilisable par les programmes, agents logiciels et machines, grâce à un système de métadonnées formelles (données servant à décrire d'autres données), utilisant notamment la famille de langages développés par le World Wide web Consortium (W3C).

La vision courante du web sémantique proposée par Berner Lee est représentée dans une architecture en plusieurs couches différentes que nous allons expliciter :

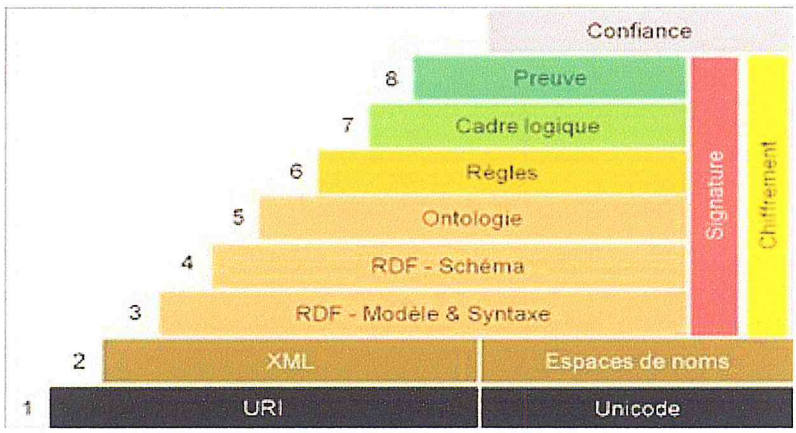

Figure 1 : pyramide du web sémantique selon Sir Tim Berners Lee[2]

- **Les couches les plus basses :** assurent l'interopérabilité syntaxique :
  - La notion d'URI (Uniform Ressource Identifier, identifiant uniforme de ressource) fournit un adressage standard universel permettant d'identifier des ressources : c'est une courte chaîne qui fournit l'identification de la ressource sur un réseau.
  - Unicode est un encodage textuel universel pour échanger des symboles.
- **XML** : est un langage proposant une syntaxe de description de la structure d'un document, mais aussi de création et de manipulation de documents. Il utilise un espace de nommage (namespace) permettant d'identifier des balises utilisées dans les documents XML. Le document est validé et définit par un schéma. Cependant, XML n'impose aucune contrainte sémantique à la signification de ces documents. XML ne suffit donc pas pour assurer aux logiciels la compréhension du contenu des données incluses dans le document : l'interopérabilité syntaxique n'est pas suffisante.
- **Les couches RDF M&S (RDF Model & Syntax) et RDF Schéma** : sont considérées comme les premières fondations de l'interopérabilité sémantique. Leur rôle essentiel est de décrire des lois de classification de concepts ou de propriétés sur des données.
  - RDF est considéré comme une fondation car c'est le premier langage fournissant un moyen d'insérer de la sémantique dans un document.
  - Le schéma RDFS décrit les hiérarchies des concepts et des relations entre les concepts.
- **La couche ontologie** : apporte une évolution car elle assure la description de sources d'information hétérogènes. Ces sources peuvent d'ailleurs formaliser une conceptualisation de choses existantes partagée par plusieurs personnes, voir par toute une communauté. Le rôle de l'ontologie est donc d'aider l'humain et la machine à communiquer, en priorisant sur l'échange de sémantique des informations plutôt que la syntaxe et en utilisant des règles précises.
- **La couche règles** : offre les moyens de l'intégration, de la dérivation, et de la transformation de données provenant de sources multiples.
- **La couche Logique** : se trouve au-dessus de la couche Ontologie. Certains les considèrent comme étant au même niveau.
- **Les couches Preuve et Confiance** : permettent de vérifier des déclarations effectuées dans le web sémantique **[3].**

Incomplétude : Les informations sur l'évènement sont incomplètes, certaines informations sont manquantes.

- **Modèles d'incertitude :**

Cette classe contient des informations sur les théories mathématiques pour les types d'incertitude. Les types spécifiques de théories comprennent, mais sans s'y limiter, ce qui suit:

- Probabilité.
- Ensembles flous.
- Fonctions de croyance.
- Combinaison de plusieurs modèles (hybrides) **[4].**

3. Les ontologies :

3.1 Historique :

Le terme Ontologie a été utilisé pour la première fois par les philosophes grecs dans une discipline qui a plus 2300 ans, qui s'intéresse à l'existence de l'être en tant qu'être et aux catégories fondamentales de l'existant. Etymologiquement parlant, Onto est un terme dérivé d'un mot grec et signifie l'Être (ainsi que ses manifestations) Logie vient du mot grec Logos qui veut dire Science ou Étude. Donc, l'Ontologie est la science qui s'intéresse à l'étude de l'être en tant qu'être.

La notion d'ontologie a été abordée dans le domaine de l'intelligence artificielle (IA) par John McCarthy. Il affirmait déjà en 1980 que les concepteurs des systèmes intelligents fondés sur la logique devraient d'abord énumérer tout ce qui existe **[5].**

3.2 Définitions :

- Dans le cadre de l'intelligence artificielle, Neeches et ses collègues **[6]** furent les premiers à proposer une définition à savoir : «une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire».

- En 1993, Gruber [7] propose la définition suivante : «spécification explicite d'une conceptualisation» qui est jusqu'à présent la définition la plus citée dans la littérature en intelligence artificielle.
- Cette définition a été modifiée légèrement par Borst [8] comme «spécification formelle d'une conceptualisation partagée».
- Ces deux définitions sont regroupées dans celle de Studer [9] comme «spécification formelle et explicite d'une conceptualisation partagée».

- Formelle : l'ontologie doit être lisible par une machine, ce qui exclut le langage naturel.
- Explicite : la définition explicite des concepts utilisés et des contraintes de leur utilisation.
- Conceptualisation : le modèle abstrait d'un phénomène du monde réel par identification des concepts clefs de ce phénomène.
- Partagée : l'ontologie n'est pas la propriété d'un individu, mais elle représente un consensus accepté par une communauté d'utilisateurs.

- Pour Guarino & Giaretta [10] «Une ontologie est une spécification rendant partiellement compte d'une conceptualisation».
- Swartout et ses collègues [11] la définissent comme suit : « une ontologie est un ensemble de termes structurés de façon hiérarchique, conçue afin de décrire un domaine et qui peut servir de charpente à une base de connaissances».
- La même notion est également développée par Gomez [12] comme: « une ontologie fournit les moyens de décrire de façon explicite la conceptualisation des connaissances représentées dans une base de connaissances. ».

Une ontologie fournit les moyens d'exprimer les concepts d'un domaine en les organisant hiérarchiquement et en définissant leurs propriétés sémantiques dans un langage de représentation des connaissances formel favorisant le partage d'une vue consensuelle sur ce domaine entre les applications informatiques qui en font usage.

3.3 Les composants d'une ontologie :

Les ontologies fournissent un vocabulaire commun d'un domaine et définissent la signification des termes et des relations entre elles, les connaissances dans les ontologies sont principalement formalisées en utilisant cinq types de composants [13] à savoir :

- **Les concepts (ou classes) :** Un concept est un constituant de la pensée (un principe, une idée, une notion abstraite) sémantiquement évaluable et communicable. L'ensemble des propriétés d'un concept constitue sa compréhension ou son intention et l'ensemble des êtres qu'il englobe son extension.
- **Les relations (ou propriétés) :** Elles représentent des interactions entre les concepts, elles permettent de construire des représentations complexes de la connaissance du domaine, elles établissent des liens sémantiques binaires, organisables hiérarchiquement.
- **Les fonctions :** Elles présentent des cas particuliers de relations dans lesquelles le nième élément de la relation est unique pour les n-1 éléments précédents. Formellement, les fonctions sont définies telles que : F : c1 * c2 * ...* cn-1*cn.
- **Les axiomes (ou règles) :** Les axiomes sont des expressions qui sont toujours vraies. Ils ont pour but de définir dans un langage logique la description des concepts et des relations permettant de représenter leur sémantique. Ils représentent les intentions des concepts et des relations du domaine et, de manière générale, les connaissances n'ayant pas un caractère strictement terminologique **[14]**. Leur inclusion dans une ontologie peut avoir plusieurs objectifs :

- Définir la signification des composants.
- Définir des restrictions sur la valeur des attributs.
- Définir les arguments d'une relation.
- Vérifier la validité des informations spécifiées ou en déduire de nouvelles.

- **Les instances (ou individus) :** Elles constituent la définition extensionnelle de l'ontologie ; elles sont utilisées pour représenter des éléments dans un domaine.

3.4 Classification des ontologies :

Les ontologies peuvent être subdivisées en plusieurs dimensions, parmi celles-ci nous en examinerons cinq, à savoir :

- Selon l'objet de la conceptualisation.
- Selon le niveau de granularité.
- Selon le niveau de formalisation de la représentation des connaissances. Selon le niveau de la représentation des connaissances.
- Selon l'information dont l'ontologie a besoin et la richesse de sa structure interne.
- Selon le niveau de la représentation des connaissances

A- **Selon l'objet de la conceptualisation :**

Dépendamment de leur objet de conceptualisation, les ontologies sont classifiées de la manière suivante :

- **Ontologie de représentation de connaissances :** Elle modélise les représentations primitives utilisées pour la formalisation des connaissances sous un paradigme donné.
- **Ontologie supérieure ou de Haut niveau** : Cette ontologie est une ontologie générale. Son sujet est l'étude des catégories des choses qui existent dans le monde, qui sont les concepts de haute abstraction tels que : les entités, les événements, les états, les processus, les actions, le temps, l'espace, les relations, les propriétés.
- **Ontologie Générique/ méta-ontologies :** Egalement appelée noyau ontologique qui modélise des connaissances moins abstraites que celles véhiculées par l'ontologie de haut niveau mais assez générales néanmoins pour être réutilisées à travers différents domaines
- **Ontologie du Domaine :** Cette ontologie exprime des conceptualisations spécifiques à un domaine, elle est réutilisable pour plusieurs applications de ce domaine. Elle fournit les concepts et les relations permettant de couvrir les vocabulaires, activités et théories de ces domaines.
- **Ontologie de Tâches :** L'ontologie de tâches fournit un vocabulaire systématisé des termes employés pour résoudre des problèmes liés aux tâches qui peuvent être ou non du même domaine.
- **Ontologie d'application :** C'est l'ontologie la plus spécifique, elle contient des concepts dépendants d'un domaine et d'une tâche particulière, elle est spécifique et non réutilisable.
- **Ontologie de méthodes :** Ce type d'ontologie modélise les définitions des concepts et des relations pertinentes pour le processus de raisonnement afin d'effectuer une tâche spécifique.

B- **Selon le niveau de granularité (détail)** : Par rapport au niveau de détail utilisé lors de la conceptualisation de l'ontologie en fonction de l'objectif opérationnel envisagé pour l'ontologie, deux catégories au moins peuvent être identifiées **[15]** :

- **Granularité fine :** ce niveau correspond à des ontologies très détaillées, elles possèdent ainsi un vocabulaire plus riche capable d'assurer une description détaillée des concepts pertinents d'un domaine ou d'une tâche.
- **Granularité large :** ce niveau correspond à des vocabulaires moins détaillés.

C- **Selon le niveau de formalisation de la représentation**: Par rapport au niveau de formalisation de la représentation des connaissances de l'ontologie, on peut distinguer quatre sortes d'ontologies:

- **Informelle :** l'ontologie est exprimée en langage naturel (sémantique ouverte). Cela peut permettre de rendre plus compréhensible l'ontologie pour l'utilisateur mais peut rendre plus difficile la vérification de l'absence de redondances ou de contradictions.
- **Semi-informelle :** l'ontologie est exprimée dans une forme restreinte et structurée du langage naturel, cela permet d'augmenter la clarté de l'ontologie tout en réduisant l'ambiguïté.
- **Semi-formelle :** l'ontologie est exprimée dans un langage artificiel défini formellement.
- **Formelle :** l'ontologie est exprimée dans un langage artificiel disposant d'une sémantique formelle ainsi que des théorèmes permettant de prouver ses propriétés, l'intérêt d'une ontologie formelle est la possibilité d'effectuer des vérifications telles que la complétude, la non-redondance, la consistance, la cohérence, etc.

D- **Selon le niveau de la représentation des connaissances :** L'appellation de niveaux de représentation des connaissances ainsi que leur interprétation peuvent varier selon les auteurs. À titre de comparaison, nous décrivons les typologies de Mizoguchi **[16]** et de Bachimont **[17],**

- d'une part, Mizoguchi propose une typologie sur trois niveaux très axée sur le processus d'ingénierie ontologique :

- **Niveau 1 (ou niveau conceptuel) :** Il est spécifié en faisant abstraction de toute contrainte informatique et sert donc de support à l'acquisition des connaissances.,
- **Niveau 2 (ou niveau formel)** : Il est formalisé au moyen d'un langage de représentation interprétable par une machine ; en plus des définitions du niveau 1, les définitions formelles sont ajoutées pour empêcher des interprétations inattendues des concepts.
- **Niveau 3 (ou niveau opérationnel) :** Il est encodé au moyen d'un langage de programmation dont la spécification obtenue est exécutable.

- D'autre part, Bachimont propose une typologie de trois niveaux très axée sur la construction de la signification :

- **Niveau sémantique (ou interprétatif) :** tous les concepts (caractérisés par un libellé) doivent respecter les quatre principes différentiels :
  - Communauté avec l'ancêtre.

- Différence (spécification) par rapport à l'ancêtre.
- Communauté avec les concepts frères (situés au même niveau).
- Différence par rapport aux concepts frères (sinon il n'aurait pas lieu de le définir).

Ces principes correspondent à l'engagement sémantique qui s'assure que le libellé de chaque concept aura un sens univoque et non contextuel associé. Deux concepts sémantiques sont identiques si l'interprétation du terme/libellé à travers les quatre principes différentiels aboutit à un sens équivalent.

- **Niveau formel (ou référentiel) :** En ajout aux caractéristiques énoncées au niveau précédent, les concepts référentiels se caractérisent par un terme/libellé dont la sémantique est définie par une extension d'objets.
- **Niveau opérationnel (ou computationnel) :** En plus des caractéristiques énoncées au niveau précédent, les concepts au niveau opérationnel sont caractérisés par les opérations qu'il est possible de leur appliquer pour générer des inférences ou engagement computationnel.

3.5 Rôles des ontologies :

Plusieurs chercheurs se sont intéressés à la finalité des exploitations des ontologies. Nous préciserons dans ce qui suit les rôles et l'intérêt des ontologies au sein des systèmes à base de connaissances (SBC) et du Web Sémantique :

• **Les connaissances du domaine d'un SBC** : Les ontologies servent à représenter les connaissances du domaine d'un SBC. En particulier, elles servent de squelette à la représentation des connaissances du domaine dans la mesure où elles décrivent les objets, leurs propriétés et la façon dont ils peuvent se combiner pour constituer des connaissances du domaine complètes.

• **La communication** : Les ontologies peuvent intervenir dans la communication entre personnes, organisations et logiciels **[18]**. En effet, les ontologies servent par exemple, à créer au sein d'un groupe ou d'une organisation un « vocabulaire conceptuel commun ». Dans ce cas, on est plutôt dans le cadre d'une ontologie informelle. Dans le cas de la communication entre personnes et systèmes. L'ontologie est un puissant moyen pour lever les ambiguïtés dans les échanges.

• **L'interopérabilité** : le développement et l'implantation d'une représentation explicite d'une compréhension partagée dans un domaine donné, peut améliorer la communication,

qui à son tour permet une plus grande réutilisation, un partage plus large et une interopérabilité plus étendue [18]. L'interopérabilité est donc une spécialisation de la communication qui permet de répertorier les concepts que des applications peuvent s'échanger même si elles sont distantes et développées sur des bases différentes.

• **L'aide à la spécification de systèmes :** La plupart des logiciels conventionnels sont construits avec une conceptualisation implicite et que la nouvelle génération des systèmes utilisant les travaux en intelligence artificielle devrait être basée sur une conceptualisation explicitement représentée.

• **L'indexation et la recherche d'information** : Dans le Web Sémantique, les ontologies y sont utilisées pour déterminer les index conceptuels décrivant les ressources sur le Web [16].

3.6 Cycle de vie d'une ontologie :

Corcho [19] insiste sur le fait que les activités de documentation et d'évaluation sont nécessaires à chaque étape du processus de construction, l'évaluation précoce permettant de limiter la propagation d'erreurs. Le processus de construction peut être intégré au cycle de vie d'une ontologie comme indiqué dans la figure suivante :

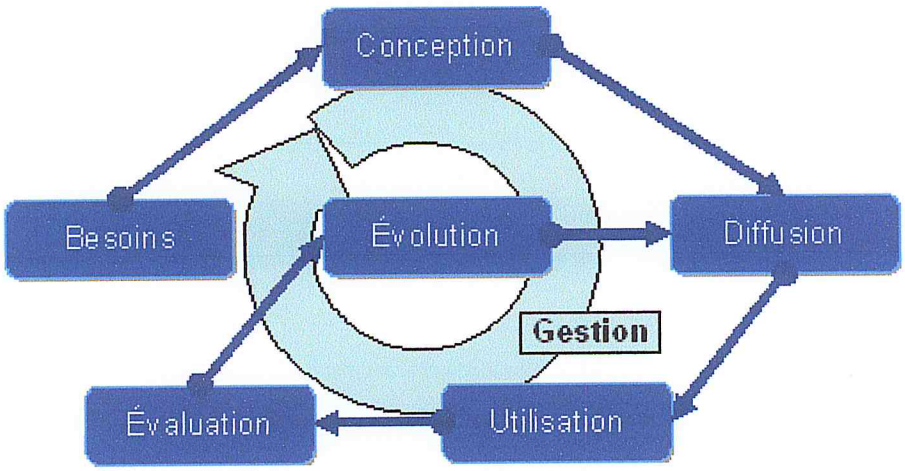

Figure 2 : cycle de vie d'une ontologie [19]

- **Besoins et évaluation :**

L'activité de détection des besoins précède la phase de conception et nécessite la mise en place de démarches méthodologiques de recueil d'information et d'identification de scénario potentiels. Cette phase de détection des besoins nécessite un état des lieux initial

approfondi dans la mesure où elle ne peut reposer sur des études précédentes ou des retours d'utilisation, comme c'est le cas pour l'évaluation.

- **Conception et évolution :**

La phase de conception de l'ontologie met en exergue un certain nombre d'actions à faire :

- Spécifier des solutions (maquettage, prototypage)
- Acquérir des connaissances nécessaires (traitement automatique de la langue naturelle, analyses de texte)
- Concevoir et modéliser (design pattern ontologiques, méta-ontologies)
- Formaliser l'ontologie (méthodes et outils de l'ontologie formelle, analyse formelle de concepts, graphes conceptuels, formalismes du web sémantique RDF/S et OWL) ;
- Intégrer des ressources existantes (alignement automatique d'ontologies, traduction) ;
- implantation (graphes conceptuels, logiques de description, formalismes objets).

Dans cette phase de conception, il s'agit également d'obtenir un consensus sur les choix de représentation et de conceptualisation de l'ontologie, car en fonction des usages, cette action nécessite des « collecticiels » et des outils de gestion de points de vue, de terminologies et des différentes langues.

Notons que la phase d'évolution de l'ontologie est liée à celle-ci car elle précède sa phase de diffusion : l'évolution pose le problème de la maintenance des contenus propres à l'ontologie existante. En effet, une ontologie est d'une part un objet vivant et également un ensemble de primitives facilitant la description des faits du domaine étudié. Lorsque l'ontologie est modifiée, les changements ont un impact sur ce qui avait été préalablement construit dans l'ontologie. La maintenance de l'ontologie soulève donc des dilemmes associés à l'intégration technique et l'intégration des usages.

- **Diffusion de l'ontologie :**

Cette phase de diffusion de l'ontologie concerne le déploiement et la mise en place de celle-ci après l'avoir créée. Elle est souvent confrontée à des problèmes de compatibilité avec les architectures des solutions utilisées. Les architectures distribuées peuvent ainsi être utilisées pour le partage de fichier, celles de services web pour l'intégration d'applications… Ces différents types d'application nécessite une harmonisation dans la distribution des ressources (données, modèles…) et leur hétérogénéité au niveau syntaxique voir sémantique, implique une recherche sur l'interopérabilité de l'ontologie diffusée.

- **Utilisation de l'ontologie :**

Cette phase du cycle de vie regroupe les activités relatives à la disponibilité de l'ontologie, par exemple, l'annotation des ressources par le biais de métadonnées, la résolution de requêtes sur cette ontologie, ou encore la déduction de connaissances et l'aide à la décision (moteurs d'inférence à base de règles). Ces types d'activités ont en commun la question de la conception des interactions avec l'utilisateur, ainsi que leur ergonomie.

- **Gestion :**

Cette phase de gestion de l'ontologie est importante car elle met en valeur l'important travail de suivi pour détecter, déclencher, préparer et évaluer les itérations du cycle de l'ontologie, en s'assurant que celle-ci reste située dans le cercle vertueux des systèmes d'information (contribution, utilisation, création) **[3].**

3.7 Les langages des ontologies :

➢ **KIF :**

KIF **[20]** est un langage basé sur les prédicats du premier ordre avec des extensions pour représenter des définitions et des méta-connaissances. Tant que la logique du premier ordre est un langage de bas niveau pour l'expression d'ontologies, l'outil Ontolingua **[21]** permet aux utilisateurs de construire des ontologies KIF à un niveau plus élevé de la description par l'importation des définitions des ontologies prédéfinies.

➢ **KL-ONE KL-ONE [23]**

Est un langage basé sur la logique de description **[23]**. Il est une formalisation de représentation de la connaissance à base de cadres (« frame-based »). Ce système maintient la définition des concepts par un simple nommage, et l'indication de la correspondance des concepts dans une hiérarchie de généralisation/spécialisation. De nouveaux termes peuvent être définis par des opérations de conjonction des concepts. De nouveaux rôles peuvent être introduits pour représenter les relations qui peuvent exister entre des individus dans le domaine modélisé. Les définitions des concepts peuvent inclure des restrictions sur les valeurs possibles, sur les nombres de valeurs, ou sur le type de valeurs qu'un rôle peut avoir pour un concept.

➢ **RDF et RDF Schéma :**

RDF **[24]** « Resource Description Framework » est un formalisme graphique pour représenter des méta-données. Il est basé sur la notion de triplet (sujet, prédicat, objet). RDF utilise la syntaxe XML, mais il ne donne aucune signification spécifique pour le vocabulaire comme sous classe de, ou le type. Les primitives de modélisation offertes par RDF sont très basiques. RDF Schéma **[24]** est un langage qui étend RDF avec un vocabulaire de termes et les relations entre ces termes. RDFS est reconnu comme un langage d'ontologie qui définit : Des classes et des propriétés. Les sous-classes, les super-classes, les sous-propriétés, et les super-propriétés. Le domaine de définition et le domaine image des propriétés.

➢ **DAML + OIL**

Beaucoup de travaux ont été faits dans le domaine de la représentation des connaissances parmi lesquels on peut citer les plus importants : SHOE, OntoBroker **[25]**, OIL **[26]**, et encore DAML + OIL **[27]** qui a remplacé DAML – ONT . DAML + OIL est un langage construit sur des normes précédentes du W3C telles que RDF et RDF Schéma, et étend ces langages avec des primitives de modélisation plus riches. DAML+OIL a été conçu à partir du langage d'ontologie DAML-ONT **[28]** en vue de combiner plusieurs composants du langage OIL **[26]**. OIL « Ontology Inference Language » est une représentation basée sur le Web, et une couche d'inférence pour des ontologies

➢ **OWL**

Développé par le groupe de travail sur le Web Sémantique du W3C, OWL peut être utilisé pour représenter explicitement les sens des termes des vocabulaires et les relations entre ces termes. OWL vise également à rendre les ressources sur le Web aisément accessibles aux processus automatisés **[29]**, d'une part en les structurant d'une façon compréhensible et standardisée, et d'autre part en leur ajoutant des méta-informations. Pour cela, OWL a des moyens plus puissants pour exprimer la signification et la sémantique que XML, RDF, et RDF-S **[24]**. De plus, OWL tient compte de l'aspect diffus des sources de connaissances et permet à l'information d'être recueillie à partir de sources distribuées, notamment en permettant la mise en relation des ontologies et l'importation des informations provenant explicitement d'autres ontologies **[30]**.

OWL a trois sous langages de plus en plus expressifs : OWL Lite, OWL DL, et OWL Full :

- **OWL Lite :** supporte les utilisateurs ayant besoin principalement d'une hiérarchie de classification et des contraintes simples (un ensemble est limité à 0 ou 1 élément, par exemple). Il a une complexité formelle inférieure à celle de OWL DL. OWL Lite supporte seulement un sous-ensemble de constructions du langage OWL.
- **OWL DL :** D'après son nom OWL DL utilise la logique de description DL . Il supporte les utilisateurs qui réclament l'expressivité maximale tout en retenant la complétude informatique (toutes les conclusions sont garanties d'être calculables), et la possibilité de décision (les calculs finiront en un temps fini). Il inclut toutes les constructions du langage OWL, qui ne peuvent être utilisées que sous certaines restrictions.
- **OWL Full :** a été défini pour les utilisateurs qui veulent une expressivité maximale et une liberté syntaxique de RDF sans des garanties informatiques. OWL Full permet à une ontologie d'augmenter la signification du vocabulaire prédéfini (RDF ou OWL). Il est peu probable que n'importe quel logiciel de raisonnement soit capable de supporter le raisonnement complet de chaque caractéristique de OWL Full. Autrement dit, en utilisant OWL Full en comparaison avec OWL DL, le support de raisonnement est moins prévisible puisque l'implémentation complète de OWL Full n'existe pas actuellement.

3.8 Domaines d'application des ontologies :

- **Système d'information :** L'intégration d'une ontologie dans un système d'information vise à réduire, voire éliminer, la confusion conceptuelle et terminologique à des points clefs du système, et à tendre vers une compréhension partagée pour améliorer la communication, le partage, l'interopérabilité et le degré de réutilisation possible, ce qui permet de déclarer formellement un certain nombre de connaissances utilisées pour caractériser les informations gérées par le système, et de se baser sur ces caractérisations et la formalisation de leur signification pour automatiser des taches de traitement de l'information. L'ontologie retrouve maintenant dans une large famille de systèmes d'information. Elle est utilisée pour :

- Décrire et traiter des ressources multimédia ;
- Assurer l'interopérabilité d'applications en réseaux ;
- Piloter des traitements automatiques de la langue naturelle ;
- Construire des solutions multilingues et interculturelles ;
- Permettre l'intégration des ressources hétérogènes d'information ;
- Vérifier la cohérence de modèles ;
- Permettre les raisonnements temporel et spatial ;
- Faire des approximations logiques ;
- etc.

Ces utilisations des ontologies se retrouvent dans de nombreux domaines d'applications tel que :

- Intégration d'information géographique ;
- Gestion de ressource humaine ;
- Aide à l'analyse en biologie, suivi médicale informatisé ;
- Commerce électronique ;
- Enseignement assisté par ordinateur ;
- Bibliothèque numériques ;
- Recherche d'informations.

4. Ontologie probabiliste :

Intuitivement, c'est une ontologie qui a des probabilités attaché aux certains de ses éléments, en général, les personnes confrontées au défi complexe de la représentation de l'incertitude dans des langages comme OWL ont tendance à commencer par écrire des probabilités sous forme d'annotations (par exemple. texte balisé décrivant quelques détails liés à un objet ou d'une propriété). Ceci est une solution palliative qui ne s'adresse qu'à une partie de l'information qui doit être représenté. Au cours des dernières décennies, des formalismes sémantiquement riches et informatiquement efficaces ont émergé pour représenter et raisonner avec la connaissance probabiliste. Annoter une ontologie standard avec des probabilités numériques ne suffit pas, vu que trop d'informations sont perdues à cause de l'absence d'un bon système de représentation qui capture les contraintes structurelles et les dépendances entre les probabilités. Une véritable ontologie probabiliste doit être capable de représenter correctement les nuances. Plus formellement:

Définition : Une ontologie probabiliste est une représentation de la connaissance formelle explicite qui exprime des connaissances sur un domaine d'application. Ceci comprend :

- Types d'entités qui existent dans le domaine;
- Propriétés de ces entités;
- Les relations entre les entités;
- Les processus et les événements qui se passent avec ces entités;
- Régularités statistiques qui caractérisent le domaine;
- Connaissances peu concluantes, ambiguës, incomplètes, peu fiables, et dissonantes liés aux entités du domaine ;
- Incertitude sur toutes les formes de connaissance citées ci-dessus ;

D'où, le terme entité se réfère à tout concept (réel ou fictif, concret ou abstrait) qui peut être décrit et raisonné sur selon le domaine d'application.

Les ontologies probabilistes sont utilisés dans le but de décrire en détail les connaissances sur un domaine et l'incertitude associée à ces connaissances d'une manière raisonnée, structurée et partageable, idéalement dans un format qui peut être lu et traité par un ordinateur. Ils élargissent également les possibilités d'ontologies standards en introduisant l'exigence d'une représentation adéquate des régularités statistiques et les preuves incertaines sur les entités dans un domaine d'application **[31]**.

5. Conclusion :

Dans ce chapitre nous avons présenté le web sémantique ainsi que les ontologies avec leurs caractéristiques, leurs constructions, leurs classification, rôle, ainsi que les domaines d'application de ces ontologies. Nous avons décrit quelques langages de représentation des ontologies utilisés, à la fin nous avons introduit la notion des ontologies probabilistes.

Dans ce qui est présenté dans ce chapitre, il ressort que la notion d'ontologie constitue l'une des approches les plus efficaces pour représenter et analyser et traiter des connaissances, et que grâce au web sémantique, l'ontologie a donc trouvé un certain formalisme à l'échelle mondiale. Ainsi, on constate que les ontologies probabilistes jouent un grand rôle pour représenter l'incertitude dans le web sémantique.

Chapitre 2 : les réseaux bayésiens

1. Introduction :

La représentation des connaissances et le raisonnement à partir de ces représentations a donné naissance à de nombreux modèles. Les modèles graphiques probabilistes, et plus précisément les réseaux bayésiens (RB), initiés par Judea Pearl dans les années 1980 **[32]**, se sont révélés des outils très pratiques pour la représentation de connaissances incertaines et le raisonnement à partir d'informations incomplètes, dans de nombreux domaines comme la bio-informatique, la gestion du risque, le marketing, la sécurité informatique, le transport, etc.

Les réseaux bayésiens reposent sur un formalisme basé sur les théories des probabilités et des graphes.

2. Définition des réseaux bayésiens :

Un réseau bayésien est un système représentant la connaissance et permettant de calculer des probabilités conditionnelles apportant des solutions à différentes sortes de problématiques.

Un réseau bayésien B = (G, θ) est défini par :

– Un graphe orienté sans circuit (DAG) G=(V,E), où V est l'ensemble des nœuds de G, et E l'ensemble des arcs de G ;

– Un espace probabilisé fini (Ω ,Z, p)

_ Ensemble des variables aléatoires associées aux nœuds du graphe et définies sur (Ω ,Z, p) tel que :

$$P(X_1, X_2, \cdots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa(X_i)) \qquad (1)$$

Où $Pa(X_i)$ est l'ensemble des causes (parents) de X_i dans le graphe G.

Un réseau bayésien est donc un graphe causal auquel on a associé une représentation probabiliste sous-jacente. Cette représentation permet de rendre quantitatifs les raisonnements sur les causalités que l'on put faire à l'intérieur du graphe **[33]** .

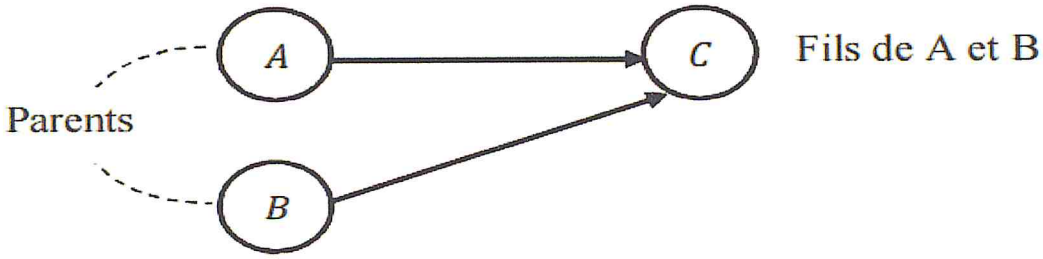

Figure 3 : Exemple d'un réseau bayésien simple

3. Pourquoi choisir les réseaux bayésiens ?

Selon le type d'application, l'utilisation pratique d'un réseau bayésien peut être envisagée au même titre que celle d'autres modèles : réseau de neurones, système expert, arbre de décision, modèle d'analyse de données (régression linéaire), arbre de défaillances, modèle logique. Naturellement, le choix de la méthode fait intervenir différents critères, comme la facilité, le coût et le délai de mise en œuvre d'une solution. En dehors de toute considération théorique, les aspects suivants des réseaux bayésiens les rendent, dans de nombreux cas, préférables à d'autres modèles :

- ✓ Acquisition des connaissances. La possibilité de rassembler et de fusionner des connaissances de diverses natures dans un même modèle : retour d'expérience (données historiques ou empiriques), expertise (exprimée sous forme de règles logiques, d'équations, de statistiques ou de probabilités subjectives), observations. Dans le monde industriel, par exemple, chacune de ces sources d'information, quoique présente, est souvent insuffisante individuellement pour fournir une représentation précise et réaliste du système analysé.

- ✓ Représentation des connaissances. La représentation graphique d'un réseau bayésien est explicite, intuitive et compréhensible par un non spécialiste, ce qui facilite à la fois la validation du modèle, ses évolutions éventuelles et surtout son utilisation. Typiquement, un décideur est beaucoup plus enclin à s'appuyer sur un modèle dont il comprend le fonctionnement qu'à faire confiance à une boîte noire.

- ✓ Utilisation des connaissances. Un réseau bayésien est polyvalent : on peut se servir du même modèle pour évaluer, prévoir, diagnostiquer, ou optimiser des décisions, ce qui contribue à rentabiliser l'effort de construction du réseau bayésien.

- ✓ Qualité de l'offre en matière de logiciels. Il existe aujourd'hui de nombreux logiciels pour saisir et traiter des réseaux bayésiens. Ces outils présentent des fonctionnalités plus ou moins évoluées : apprentissage des probabilités, apprentissage de la structure du réseau bayésien, possibilité d'intégrer des variables continues, des variables d'utilité et de décision, etc **[34].**

4. Apprentissage dans les réseaux bayésiens :

L'apprentissage des réseaux bayésiens doit répondre aux deux questions suivantes :

- Comment estimer les lois de probabilités conditionnelles ?
- Comment trouver la structure du réseau bayésien ?

Nous allons donc séparer le problème de l'apprentissage en deux parties :

- *L'apprentissage de* paramètres, où nous supposerons que la structure du réseau a été fixée, et où il faudra estimer les probabilités conditionnelles de chaque nœud du réseau.
- *L'apprentissage de la structure*, où le but est de trouver le meilleur graphe représentant la tâche à résoudre.

4.1 Apprentissage des paramètres :

4.1.1 À partir des données complètes:

Dans le cas où toutes les variables sont observées, la méthode la plus simple et la plus utilisée est l'estimation statistique qui consiste à estimer la probabilité d'un événement par la fréquence d'apparition de l'événement dans la base de données.

4.1.2 À partir des données incomplètes :

Dans les applications pratiques, les bases de données sont très souvent incomplètes. Certaines variables ne sont observées que partiellement ou même jamais, que ce soit en raison d'une panne de capteurs, d'une variable mesurable seulement dans un contexte bien précis, d'une personne sondée ayant oublié de répondre à une question, etc **[35].**

- ➢ **L'algorithme EM pour l'apprentissage des données incomplètes :**

L'algorithme EM — pour Expectation-Maximisation — est un algorithme itératif du à Dempster, Laird et Rubin (1977). Il s'agit d'une méthode d'estimation paramétrique s'inscrivant dans le cadre général du maximum de vraisemblance. Lorsque les seules données dont on dispose ne permettent pas l'estimation des paramètres, et/ou que

l'expression de la vraisemblance est analytiquement impossible à maximiser, l'algorithme EM peut être une solution. De manière grossière et vague, il vise à fournir un estimateur lorsque cette impossibilité provient de la présence de données cachées ou manquantes ou plutôt, lorsque la connaissance de ces données rendrait possible l'estimation des paramètres. L'algorithme EM tire son nom du fait qu'à chaque itération il opère deux étapes distinctes :

- la phase « Expectation », souvent désignée comme *« l'étape E »*, procède comme son nom le laisse supposer à l'estimation des données inconnues, sachant les données observées et la valeur des paramètres déterminée à l'itération précédente ;
- la phase « Maximisation », ou « étape M », procède donc à la maximisation de la vraisemblance, rendue désormais possible en utilisant l'estimation des données inconnues effectuée à l'étape précédente, et met à jour la valeur du ou des paramètre(s) pour la prochaine itération **[36]** .

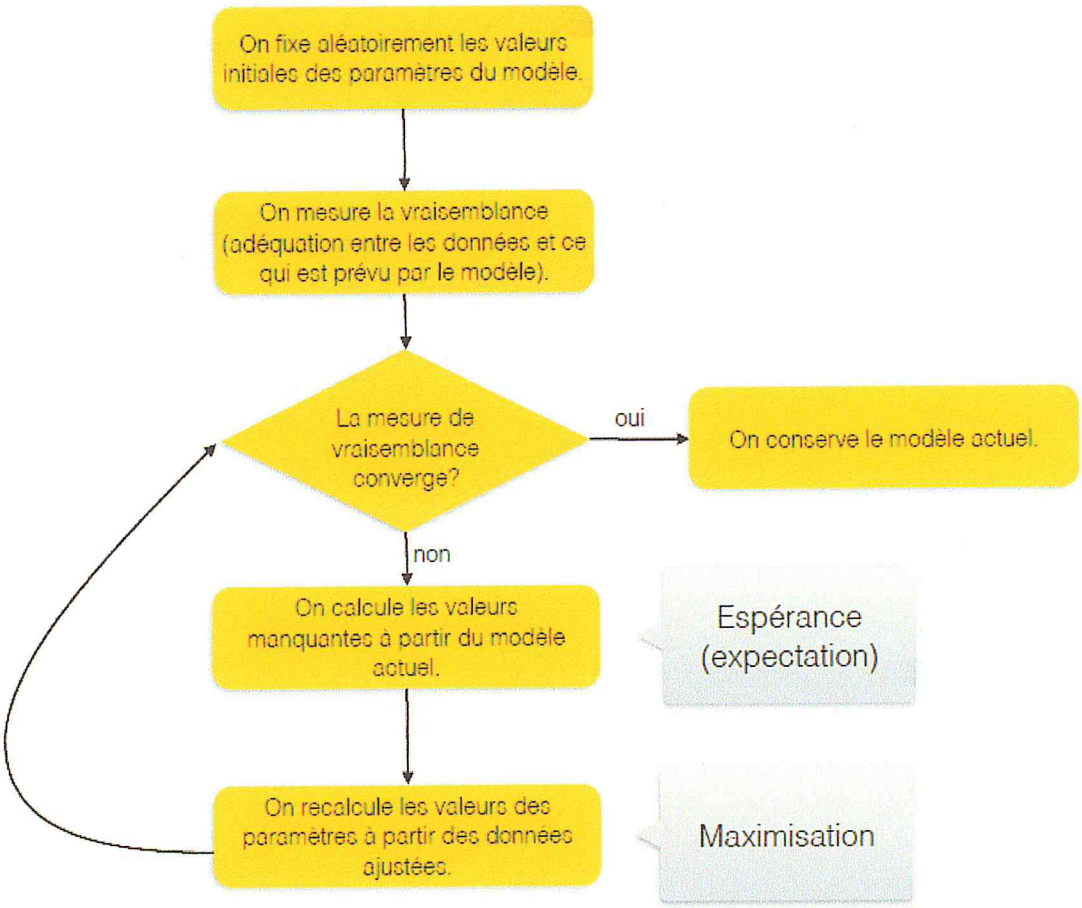

Figure 4 : L'algorithme EM pour le maximum de vraisemblance

4.2 Apprentissage de la structure :

L'objectif de l'apprentissage de la structure est de trouver une structure du graphe à partir des données disponibles et qui représente le mieux un problème. Une solution naïve pour trouver la meilleure structure d'un réseau bayésien, est de parcourir tous les graphes possibles, de leur associer un score, puis de choisir le graphe qui a le score le plus élevé. Cependant, le nombre de structures différentes pour un réseau bayésien de n nœuds est super exponentiel. Il est donc impossible d'effectuer un parcours exhaustif en un temps raisonnable. Pour cette raison, la plupart des méthodes d'apprentissage de structure utilisent une heuristique de recherche dans l'espace des graphes acycliques dirigés.

Les méthodes d'apprentissage de structure qui se basent sur un calcul de score maximisent le score de la structure G qui décrit le mieux les données D.

Par exemple, Score (G, D) = P(G|D) = P(D|G).P(G) P(D) .

L'approche bayésienne de l'apprentissage de structure se résume à deux éléments essentiels :

La fonction d'évaluation associée à une structure, et la procédure de recherche de la structure visant à maximiser la fonction d'évaluation.

Il existe divers algorithme pour l'apprentissage de la structure, y compris l'algorithme PC, recherche de causalité, l'arbre de poids minimum, la recherche gloutonne, l'algorithme Structural-EM, et l'algorithme K2.

- **Algorithme K2 :**

  Est une méthode efficace a été proposée par Cooper & Hersovits(1992). Cette méthode nommée K2 est à présent largement utilisée bien qu'elle possède l'inconvénient de demander un ordre d'énumération en paramètre d'entrée. L'idée de l'algorithme K2 est de maximiser la probabilité de la structure sachant les données dans l'espace de DAG respectant cet ordre d'énumération.

  L'algorithme K2 teste alors l'ajout de parents de la manière suivante : le premier nœud ne peut pas posséder de parents et pour les nœuds suivants les parents choisis sont le sous-ensemble de nœuds qui augmente le plus le score parmi l'ensemble des nœuds le précédent dans l'ordre d'énumération. L'espace de recherche est ainsi réduit à l'ensemble des DAG respectant cet ordre **[37].**

  La figure suivante montre l'algorithme K2 :

ENTRÉES: Un ensemble de n sommets, un ordre total sur ces sommets, une borne supérieure u au nombre de parents qu'un sommet peut avoir, un ensemble D contenant m exemples.

SORTIES: Pour chaque sommet, l'ensemble des parents de ce sommet.

pour $i = 1$ à n **faire**
  $\Pi_i \leftarrow \emptyset$
  $P_{old} \leftarrow g(i, \Pi_i)$
  $Continuer \leftarrow VRAI$
  tantque $Continuer$ ET $|\Pi_i| < u$ **faire**
    $Z \leftarrow$ sommet de $Pred(X_i) - \Pi_i$ qui maximise $g(i, \Pi_i \cup Z)$
    $P_{new} \leftarrow g(i, \Pi_i \cup Z)$
    si $P_{new} > P_{old}$ **alors**
      $P_{old} \leftarrow P_{new}$
      $\Pi_i \leftarrow \Pi_i \cup Z$
    sinon
      $Continuer \leftarrow FAUX$
    finsi
  fin tantque
fin pour
Retourner Π.

- Π_i : ensemble des parents du sommet x_i;
- ϕ_i : liste de toutes les instanciations possibles de x_i dans D;
- $q_i = |\phi_i|$;
- r_i : nombre de modalités de l'attribut x_i;
- α_{ijk} : nombre d'exemples de D pour lesquels l'attribut x_i est instancié à sa k^{me} valeur, et ses parents à leur j^{eme} valeur dans ϕ_i;
- $N_{ij} = \sum_{k=1}^{r_i}(\alpha_{ijk})$, c'est-à-dire le nombre d'exemples de D pour lesquels les parents de x_i sont instanciés à leur j^{eme} valeur dans ϕ_i.

Figure 5 : L'algorithme K2[38]

5. L'inférence dans les réseaux bayésiens

L'inférence dans un réseau bayésien concerne le calcul de la probabilité de n'importe quelle variable ou sous ensemble de variables à partir des autres variables observées. Il s'agit donc de déterminer les probabilités conditionnelles d'événements reliés par des relations d'influences.

Les algorithmes d'inférence dans les réseaux bayésiens se répartissent en deux groupes : algorithmes *d'inférence exacte* et algorithmes *d'inférence approchée*. Les algorithmes d'inférence exacte exploitent les indépendances conditionnelles contenues dans le réseau pour calculer les probabilités a posteriori exactes **[32],[35].** Concernant la deuxième catégorie d'algorithmes, les méthodes utilisées donnent des estimations approchées des probabilités à posteriori **[36], [39].**

5.1 Inférence exacte :

La tâche de base de tout système d'inférence probabiliste consiste à calculer la distribution de probabilités a posteriori d'un ensemble de **variables de requête**, étant donné un **événement** observé _ autrement dite, une affectation de valeurs à un ensemble de **variables d'observation.**

X représente la variable de requête ; **E** représente l'ensemble de variable d'observation E_1.....E_m, et **e** est un événement observé particulier ; **Y** les variables non observées Y_1.....Y_l (parfois nommées **variables cachées**). D'où l'ensemble complet de variables **X = {X} U E U Y.**

Une requête de type demande la distribution de probabilité à posteriori : **P(X|e) [40].**

5.2 Inférence approchée :

Etant donné que l'inférence exacte est impraticable dans de grands réseaux multiplement connectés, il est essentiel d'envisager des méthodes d'inférence approchée. Cette section décrit d'algorithme d'échantillonnage aléatoire, également nommés algorithme de Monte-Carlo, qui fournissent des réponses approchées dont l'exactitude dépend du nombre d'échantillons générés. Ces dernières années, les algorithmes de Monte-Carlo ont été largement utilisés en informatique pour estimer des quantités difficiles à calculer exactement.

Dans cette partie nous nous intéressons à l'échantillonnage appliqué au calcul de probabilité à posteriori nous décrivons deux familles d'algorithmes : l'échantillonnage direct, l'échantillonnage par rejet **[41]**.

6. Les réseaux bayésiens multi entités (MEBN) :

MEBN (Multy Entity Bayesian Network) est un système logique qui intègre la logique du premier ordre (First Order Logic (FOL)) avec la théorie des probabilités bayésienne, il étend les réseaux bayésiens ordinaires pour permettre la représentation des modèles graphiques avec des sous-structures répétées. La connaissance est codée comme une collection de fragments de réseaux bayésiens (MFrags) qui peuvent être instanciés et combinés pour former des réseaux bayésiens très complexes spécifiques à chaque situation. Une théorie MEBN (MTheory) représente implicitement une distribution de probabilité conjointe sur les chiffres éventuellement non bornées d'hypothèses et utilise l'apprentissage bayésien pour affiner une base de connaissances quand les observations s'accumulent. MEBN fournit une base logique pour la collection émergente des langues à base de probabilités très expressives **[42]**.

Un MFrag représente la connaissance incertaine d'une collection de variables aléatoires connexes (VAs), les VAs également connu sous le nom des «nœuds» d'un MFrag, représentent les attributs et les propriétés d'un ensemble d'entités. Un graphe orienté représente les dépendances entre les VAs. Leurs variables aléatoires contiennent habituellement comme arguments une ou plusieurs variables ordinaires, qui sont des variables qui sont substituées par des instances d'entités au cours du processus d'instanciation. **[43]**.

- **Définition des nœuds (variables aléatoires) :**

  1- Nœud résidant (resident node): ce sont les variables aléatoires réelles qui forment le cœur du sujet d'un MFrag. La logique MEBN exige que la distribution probabiliste locale de chaque nœud résident doit être unique et explicitement défini dans son MFrag original. Les valeurs possibles d'un nœud résident peuvent être une entité existante.

  2- Nœuds d'entrée (input node) : ces nœuds sont essentiellement des «pointeurs» faisant référence aux nœuds résidents d'un autre MFrag. Les nœuds d'entrée fournissent également un mécanisme permettant la réutilisation des nœuds résidents entre les MFrags. Les nœuds d'entrée influencent la distribution de probabilité des nœuds résidents qui sont leurs enfants dans un MFrag donné, mais leurs propres distributions sont définis ailleurs (c'est à dire dans leurs propres MFrags

d'origine). Dans une MTheory complète, chaque nœud d'entrée doit pointer vers un nœud résident dans un MFrag.

3- **Nœuds de contexte (context node) :** ce sont des variables aléatoires booléennes représentant des conditions qui doivent être satisfaites pour faire une distribution dans un MFrag valide. Les nœuds de contexte peuvent représenter plusieurs types de modèles d'incertitude sophistiqués, tels que l'incertitude sur les relations entre les entités. Si on peut inférer de la base de connaissances (c'est à dire l'ontologie) que la valeur d'un nœud de contexte est vraie, la distribution de probabilité de l'MFrag sera appliquée dans le modèle d'inférence. Si cette valeur est fausse, une distribution par défaut sera utilisée. Si la valeur est inconnue, le nœud de contexte devient pratiquement un parent de tous les nœuds résidents dans le même MFrag **[43].**

7. Quelques domaines d'application :

Il existe plusieurs domaines d'application des réseaux bayésiens dont :

- **Santé :**

  Les premières applications des réseaux bayésiens ont été développées dans le domaine du diagnostic médical.

  Les réseaux bayésiens sont particulièrement adaptés à ce domaine parce qu'ils offrent la possibilité d'intégrer des sources de connaissances hétérogènes (expertise humain et données statiques), et surtout parce que leur capacité à traiter des requêtes complexes (explication la probable, action la plus appropriée) peuvent constituer une aide véritable et interactive pour le praticien.

  Dans le domaine de la santé, une application intéressante des algorithmes issus des réseaux bayésiens a permis d'améliorer considérablement la recherche de la localisation de certains gènes, dans le cadre du projets Human Genom **[33].**

- **Informatique :**

  Les réseaux bayésiens sont probablement l'une des technologies les plus adaptées pour construire l'intelligence des agents. Ils assurent en effet ses différentes propriétés :

- L'autonomie : représentée par la capacité des réseaux bayésiens de fournir des décisions en présence d'incertitude, ou en l'absence de certaines informations.

- La motivation : peut être représentée par certains types d'inférences, ou par un système de planification.
- La réactivité : c'est le principe même de l'inférence dans les réseaux bayésiens.
- L'adaptation à l'environnement : elle est rendu possible par les capacités d'apprentissage incrémental des réseaux bayésiens.

La compacité de la représentation de la connaissance autorisée par les réseaux bayésiens est aussi un avantage pour en faire une intelligence embarquée.

L'utilisation des réseaux bayésiens dans les agents bureautiques a été largement développée par Microsoft dans les outils d'aide et de diagnostic pour son système d'exploitation Windows, à partir de Windows 98. De même, l'agent Office Assistant est un system d'aide proactif intégré dans Office, à partir de la version 97. Plusieurs agents de support technique de Microsoft ont également été développés dans le cadre du projet LUMIERE du groupe DTAS (Decision Theory and Adaptive Systems).

L'application Vista, peut également être considérée comme un agent intelligent, dont le rôle est de sélectionner les données présentées à un utilisateur en fonction de l'état du système physique qu'il doit superviser **[33]**.

- **Gestion de connaissance :**

  Le domaine de la gestion des connaissances, qui connait un intérêt croissant, est également un champ d'application potentiel pour les réseaux bayésiens, dans la mesure où ceux-ci offrent un formalisme riche et intuitif de représentation de la connaissance **[33]**.

8. Conclusion :

Dans ce chapitre nous avons présenté les réseaux bayésiens et nous avons passé en revue quelques algorithmes d'apprentissage ainsi que les méthodes d'inférence les plus connues, on a parlé des réseaux bayésiens multi entités (MEBN), par la suite nous avons cité quelques domaines d'applications ainsi que les outils des réseaux bayésiens les plus utilisés. Les réseaux bayésiens sont donc un outil de choix dans la représentation de connaissances et dans l'exploitation de celles-ci.

Chapitre 3 : langage probabiliste à base des réseaux bayésiens

1. Introduction :

Les ontologies traditionnelles (formelles) n'ont pas de mécanisme intégré pour représenter ou inférer avec l'incertitude. PR-OWL, qui s'adresse à cette lacune, se compose d'un ensemble de classes, sous-classes et propriétés qui forment collectivement un Framework pour le raisonnement avec les ontologies probabilistes. Il existe trois langages pour la représentation de l'incertitude dans les ontologies, BAYESOWL, ONTOBAYES, et le PR-OWL. Nous avons choisi le PR-OWL pour la modélisation car il est le plus courent et grâce à son intégration de la logique du premier ordre avec les réseaux bayésiens, il peut traiter les cas les plus complexe.

2. BayesOWL :

fournit un ensemble de règles et procédures pour la traduction directe d'une ontologie OWL en un RB (graphe acyclique dirigé) et une méthode basée sur des procédures d'ajustement proportionnel itératif qui incorpore des contraintes de la probabilité disponible lors de la construction des tables de probabilités conditionnelles du RB. Le RB traduit, qui préserve la sémantique de l'ontologie originale et qui est compatible avec toutes les contraintes de probabilité donnée, peut soutenir le raisonnement sur des ontologies comme inférences bayésienne **[44]**.

3. Ontobayes :

Une extension de OWL qui porte des annotations de probabilité et de dépendance pour construire des RB depuis des ontologies, ça permet le traitement des variables aléatoires multiples d'une valeur dans la construction de la table de probabilité conditionnelle, l'approche nécessite encore l'extension de l'ontologie avec les constructeurs de l'ontologie probabiliste**[45].**

4. PR-OWL

PR-OWL est une extension qui permet à une ontologie OWL de représenter des modèles probabilistes bayésiens complexes d'une manière suffisamment flexible pour être utilisé par divers outils probabilistes bayésiens (exemple : Netica, Hugin, Quiddity*Suite, JavaBayes, etc.) **[1]**.

Ce niveau de flexibilité ne peut être obtenue qu'en utilisant la sémantique sous-jacentes de la logique bayésienne du premier ordre, qui ne fait pas partie de la sémantique OWL standard et de la syntaxe abstraite. Par conséquent, il semble clair que PR-OWL ne peut être réalisé que par l'extension de la sémantique et de la syntaxe abstraite d'OWL.

PR-OWL fait donc une représentation formelle et explicite des connaissances du domaine étudié. Outre les différentes caractéristiques d'OWL, PR-OWL tient compte des régularités statistiques des champs et conclut sur de nouvelles formes de connaissance par inférence tout en tenant compte de l'incertitude autour de ces connaissances **[46].**

PR-OWL utilise une autre forme de combinaison de classes, au lieu de (<sujet><prédicat><objet>) d'OWL, et les relations qui les relient ensemble.

La figure 2 présente les classes dans un ovale et les relations qu'elles entretiennent ensemble sont indiquées par des flèches.

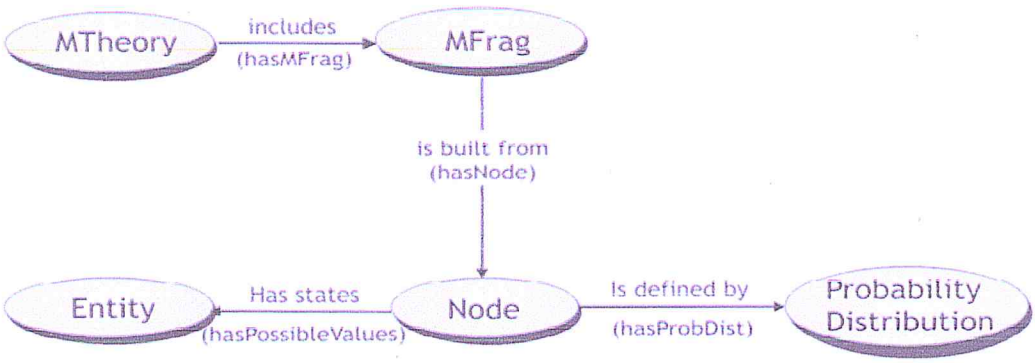

Figure 6 : Eléments principaux de PR-OWL vue globale [47]

PR-OWL est basé sur la logique du premier ordre bayésienne appelé Multi-Entités réseaux bayésiens (Multi-Entity Bayesian Networks (MEBN)) **[48].**

De manière générale, la classe MTheory représente un concept du monde, pour lequel l'ontologie va être construite. Un MTheory est un regroupement de MFrag qui correspond à des sous-classes ou des propriétés de classes destinées à acquérir de nouvelles connaissances. Un fragment est constitué d'un ensemble de variables de contexte, d'un ensemble de variables d'entrée, d'un ensemble de variables résidentes, d'un DAG sur les variables d'entrée et les

variables résidentes (dans lequel les variables d'entrée sont des nœuds racines) et d'un ensemble de distributions conditionnelles locales pour chaque variables résidentes **[49]** .

Chaque nœud a des états qui lui sont propres appelé des entités (Entity); les nœuds sont définis par une table de distribution des probabilités conditionnelles (TPC) qui définit la connaissance apriori du modèle. Les MFrags calculent la probabilité jointe de chacune de ses variables aléatoires tant dis que PR-OWL utilise les informations encapsulées dans les Mfrags pour répondre aux requêtes probabilistes **[50].**

4.1 La différence entre OWL et PROWL :

Le tableau ci-dessous montre la différence entre une ontologie formelle OWL et une ontologie probabiliste PR-OWL:

OWL	PR-OWL
Standardisé	Non standardisé
Langage déterministe	Langage probabiliste
Base logique	Réseau bayésien
Compatible : XML, RDF	Compatible : XML, RDF
Ne tient pas compte de l'incertain	Tient compte de l'incertain
Formalisme : Sujet, prédicat, objet	MTherory, MFrag
Définir les classes et les taxonomies	Définir les entités et les MFrags
Spécification des propriétés des classes, déterminer les restrictions	Définir les variables aléatoires, les variables ordinaires, et les restrictions (nœuds de contexte)
Ajouter les individus à l'ontologie et remplir les valeurs des propriétés	Ajouter les instances aux entités et remplir les preuves des variables aléatoires
Les classes sont sous la classe Thing	Les classes sont sous Entity (aussi sous Thing)
Les composants des assertions (ABox)	Individus (instances de la classe Entity) et découvertes

Tableau 1 : La différence entre OWL et PR-OWL

4.2 Les concepts PR-OWL :

Pour construire les concepts spécifiques à un domaine étudié donné, on utilise les définitions PR-OWL pour représenter l'incertitude sur leurs attributs et relations.

La figure ci-dessous représente les différents composants du PR-OWL :

Figure 7 : La hiérarchie des classes de PROWL [51].

4.2.1 Définition des classes de PR-OWL:

⇨ **RandomVariables (Les variables aléatoires):**

Une variable aléatoire en probabilités et statistiques, est une fonction qui mappe les éléments d'un espace d'échantillon à des nombres réels, L'espace de l'échantillon représente tous les résultats possibles d'un événement ou d'une expérience. La valeur de la variable aléatoire est inconnue avant que l'événement se produise. Même si le résultat est mappé à un nombre réel, le nombre réel peut être utilisé pour représenter des valeurs catégoriques, des valeurs booléennes, et d'autres types de valeurs différentes que les nombres réels tant qu'elles représentent des résultats mutuellement exclusifs et collectivement exhaustifs.

La figure ci-dessous montre les principaux concepts pour définir les variables aléatoires :

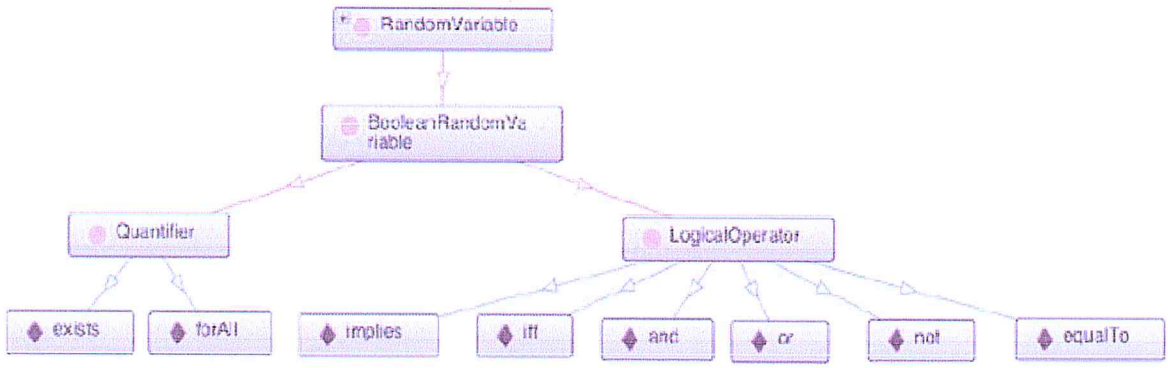

Figure 8 : Graphe avec les principaux concepts pour définir des variables aléatoires.

Dans PR-OWL, une variable aléatoire définit l'incertitude du résultat lié à une propriété spécifique, qui a sa sémantique définies dans OWL. Il existe quatre principaux concepts qui doivent être définis pour chaque variable aléatoire :

- Un lien vers la propriété OWL qui « définit l'incertitude de » représenté par la propriété *prowl2:definesUncertaintyOf*,
- Le domaine de la variable aléatoire définie par ses arguments représenté par la propriété *prowl2:hasArgument*,
- Le rang ou les résultats possibles de la variable aléatoire représenté par la propriété *prowl2:hasPossibleValues*,
- Et enfin, sa distribution représenté par la propriété *prowl2:hasProbabilityDistribution*.

En MEBN, chaque variable aléatoire a 'absurde' comme une de ses valeurs possibles. Par conséquent, dans PR-OWL cela est également le cas.

Pour dire qu'une variable aléatoire définit l'incertitude de certaines propriétés, il faudrait définir la classe *RandomVariable* ayant la restriction *definesUncertaintyOf* de certaines propriétés rdf *rdf: Property* **[51].**

- **BooleanRandomVariable** : est un type spécial des variables aléatoires, qui représente des variables aléatoires qui ne peuvent avoir que des valeurs booléennes comme leurs valeurs possibles ou rang, garanti par la restriction *hasPossibleValues.*
  - **LogicalOperator :** ou operateur logique, est un type spécial des variables aléatoires booléennes (*BooleanRandomVariable*) qui représente les opérateurs de la logique du premier ordre (FOL), ils sont principalement utilisés pour exprimer des formules de FOL utilisant des expressions de MEBN.
  - **Quantifier :** ou quantificateur, qui représente un quantificateur du premier ordre. Ils sont principalement utilisés pour exprimer des formules de la logique du premier ordre utilisant des expressions de MEBN **[51]**.

⇨ **MTheory :**

C'est une collection de fragments de MEBN (MFrags) qui satisfait les contraintes de consistance assurant l'existence d'une distribution jointe sur les variables aléatoires mentionnées dans la théorie de MEBN. Dans PR-OWL, la classe MTheory permet à une ontologie probabiliste d'avoir plus d'une théorie valide pour représenter ses variables aléatoires **[51]**.

⇨ **MFrag :**

C'est la structure de base d'un modèle logique de MEBN, il représente des influences entre les groupes de variables aléatoires liées. Pour représenter les modèles, MEBN utilise des variables ordinaires, qui sont des variables libres dans des formules ou des termes qui peuvent être substitués par des identificateurs d'entité, et / ou des exemplaires, qui correspondent à des variables liées dans les formules quantifiées. Dans PR-OWL, chaque sous-classe de la classe MFrag représente un fragment MEBN. Chaque MFrag possède au moins un nœud (au moins un nœud résident). Cependant, il peut aussi avoir des nœuds d'entrée, les variables ordinaires et des exemplaires.

Un MFrag est soit un MFrag de domaine (Domain MFrag) ou un MFrag de découverte (Finding MFrag) :

- **DomainMFrag :** est la sous-classe de le classe MFrag qui comprend tous les MFrags du domaine étudié. Elle est disjointe de la classe *FindingMFrag.* Tous les MFrags générative créée par l'ingénieur de l'ontologie (i.e l'expert du domaine) sont des individus de cette classe. Un MFrag de domaine diffère d'un MFrag en permettant l'utilisation des nœuds de contexte. qui définissent les conditions nécessaires pour les relations et les distributions définies sur le MFrag pour qu'il soit valide.

  De plus, il accepte uniquement les nœuds résidents de type *DomainResidentNode* (nœud résident de domaine) et les nœuds d'entrée de type *GenerativeInputNode* (nœud d'entrée génératif).

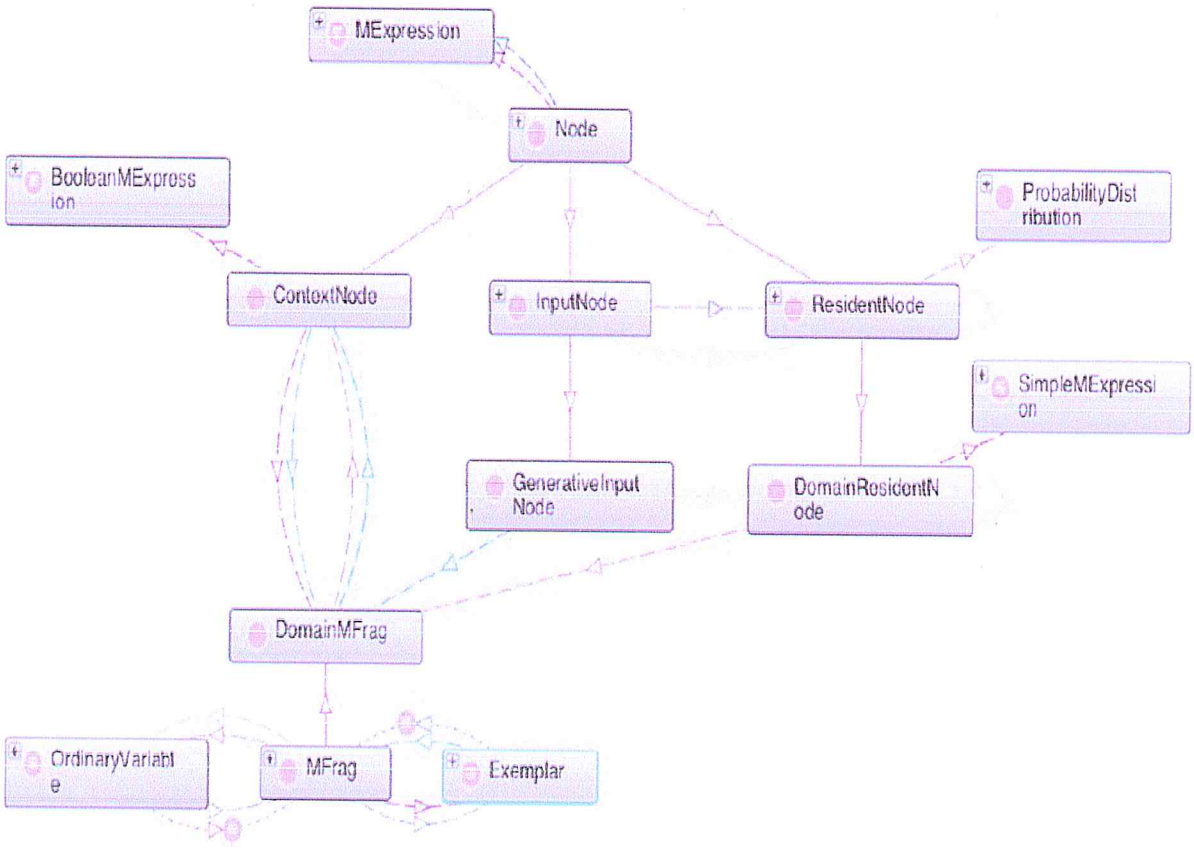

Figure 9 : Graphe avec les principaux concepts nécessaires pour définir un MFrag de domaine (DomainMFrag).

- **FindingMFrag :** est utilisé pour transmettre des informations sur les découvertes, qui sont le moyen d'entrer des preuves dans une théorie de MEBN par défaut afin qu'un algorithme probabiliste puisse être appliqué pour effectuer des inférences concernant

les nouvelles preuves. Il n'a pas de nœud de contexte, qu'un seul nœud d'entrée et un seul nœud résident **[51]**.

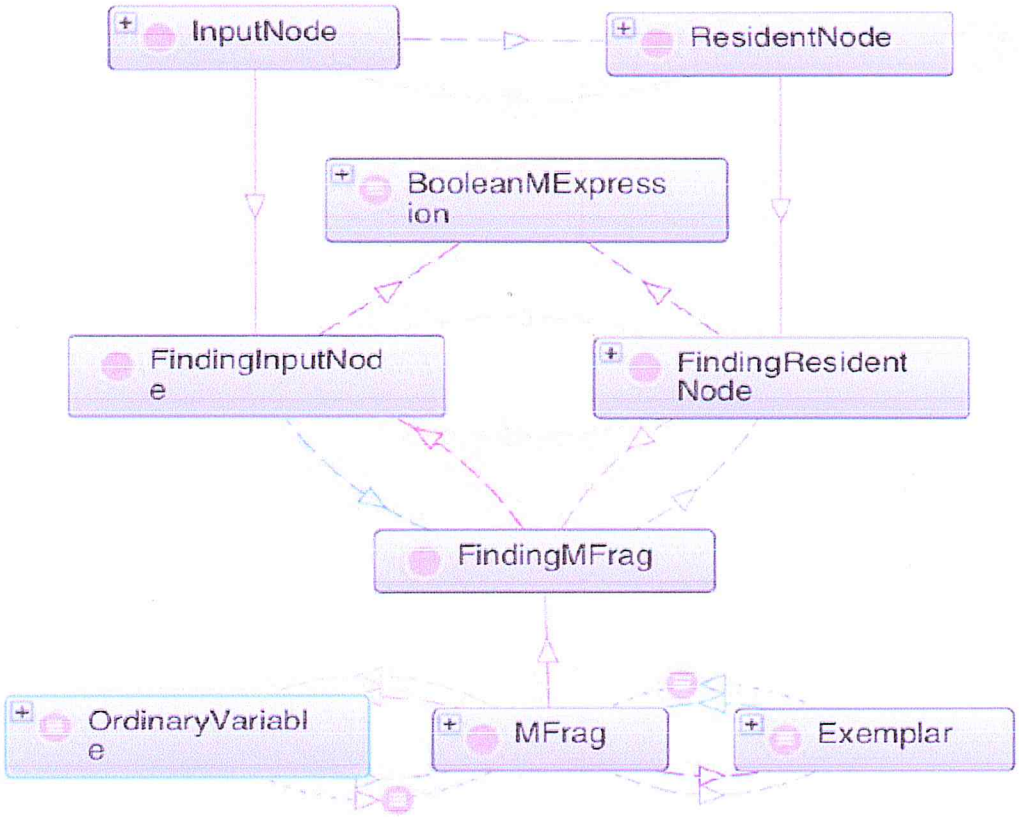

Figure 10 : Graphe avec les principaux concepts nécessaires pour définir un FindingMFrag.

⇨ **Node (nœud) :**

Il fait partie d'un MFrag et il peut définir la distribution d'une variable aléatoire au sein de ce MFrag, (un nœud résident, représenté par la classe *ResidentNode*), une variable aléatoire qui influence la répartition des nœuds dans ce MFrag mais a sa distribution défini ailleurs (un nœud d'entrée, représenté par la classe *Inputnode)* , ou une variable aléatoire qui exprime le contexte dans lequel les distributions de probabilités au sein de l'MFrag sont valides (un nœud de contexte, représenté par la classe *Contextnode*).

Les nœuds résidents, les nœuds d'entrées, et les nœuds de contextes sont des classes disjointes. Dans tous les cas, la variable aléatoire représentée par un nœud est définie comme étant une expression de MEBN.

En outre, chaque nœud réside dans exactement un seul MFrag **[51]**.

➢ **ResidentNode (nœud résident) :**

Il est associé à une variable aléatoire et il a sa distribution de probabilité définie au sein de l'MFrag. En plus de sa distribution, un nœud résident peut avoir des parents (soit nœud d'entrée ou d'un autre nœud résident) et il est un nœud résident dans exactement un MFrag.

Le nœud résident est disjointe du nœud d'entrée et du nœud de contexte.

- **DomainResidentNode** : est la sous-classe de *ResidentNode* qui comprend tous les nœuds résidents spécifiques au domaine étudié. Un nœud résident de domaine ne définit qu'une distribution sur une expression simple de MEBN (*SimpleMExpression*).

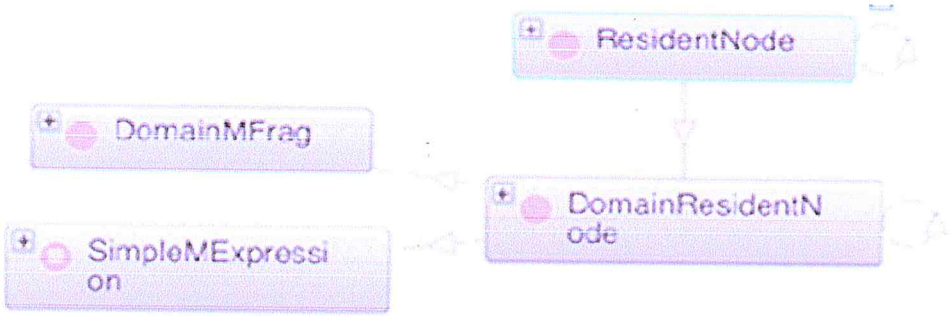

Figure 11 : Graphe avec les principaux concepts nécessaires pour définir un nœud résident du domaine (DomainResidentNode).

- **FindingResidentNode :** est la sous-classe de *ResidentNode* qui inclut tous les nœuds de découverte *(Finding nodes*). Un *FindingResidentNode* ne peut représenter qu'une expression booléenne de MEBN **[51].**

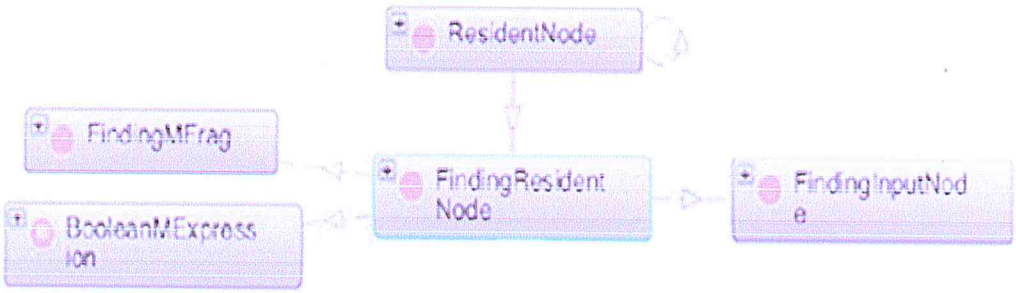

Figure 12 : Graphe avec les principaux concepts nécessaires pour définir un nœud résident de découverte (FindingResidentNode).

➢ **ContextNode (nœud de contexte) :**

Il définit une contrainte sur le MFrag là où le nœud de contexte est défini. Souvent, ces contraintes définissent le type d'arguments utilisés pour les nœuds résident.

Un nœud de contexte ne peut représenter qu'une expression booléenne de MEBN, les contraintes doivent être vraie (valide). Enfin, il ne peut être un nœud de contexte que dans exactement un MFrag de domaine **[51].**

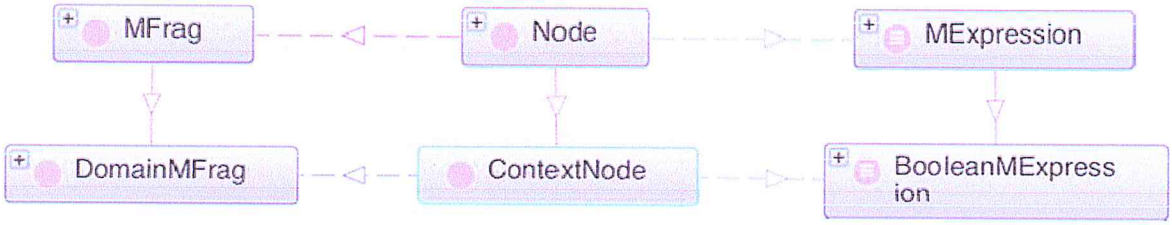

Figure 13 : Graphe avec les principaux concepts nécessaires pour définir un nœud de contexte.

➢ **InputNode (nœud d'entrée) :**

C'est une variable aléatoire qui a sa distribution définie quelque part d'autre, mais sa valeur influence sur quelques nœuds résidents dans ce MFrag. Par conséquent, il doit être un parent d'un nœud résident et il ne peut être parent qu'à des nœuds résidents. Enfin, il ne peut être un nœud d'entrée que dans exactement un MFrag de domaine.

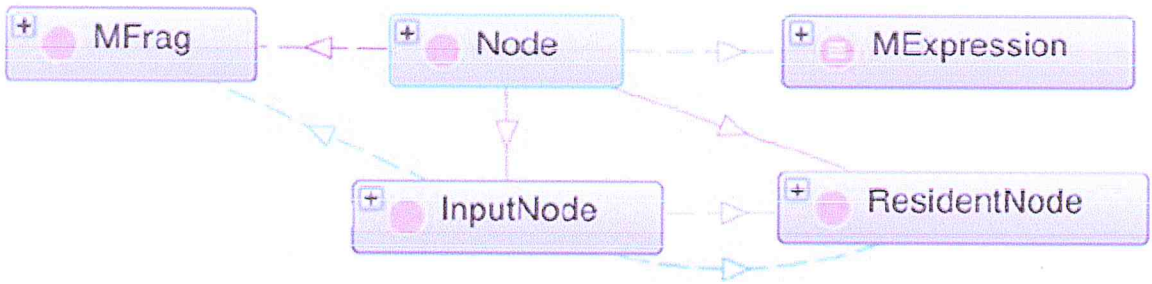

Figure 14 : Graphe avec les principaux concepts nécessaires pour définir un nœud de d'entrée.

- **FindingInputNode :** représente une expression booléenne de MEBN, qui influence quelque *FindingResidentNode* au sein du MFrag. En fait, il ne peut être un parent qu'à un seul nœud, ce nœud doit être de type *FindingResidentNode*.

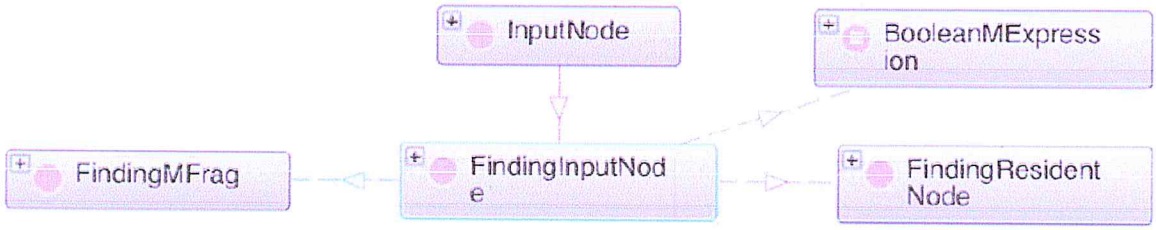

Figure 15 : Graphe avec les principaux concepts nécessaires pour définir un nœud de d'entrée de découverte.

- **GenerativeInputNode :** est une variable aléatoire qui a sa distribution défini ailleurs, mais sa valeur influence quelque *DomainResidentNode* au sein du MFrag. Il ne peut être un nœud d'entrée que dans exactement un MFrag de domaine **[51].**

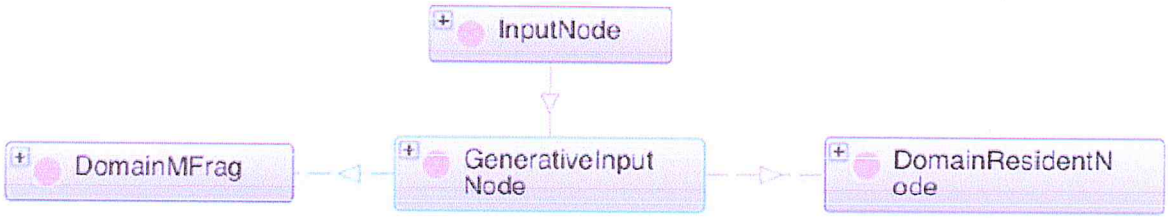

Figure 16 : Graphe avec les principaux concepts nécessaires pour définir un nœud de d'entrée génératif.

⇨ **ProbabilityDistribution :**

C'est utilisée pour définir les distributions locales pour chaque nœud résident, (Ces distributions locales s'appliquent uniquement si tous les nœuds de contexte dans le MFrag sont satisfaits), et la distribution par défaut pour une variable aléatoire (à utiliser si aucunes distributions locales dans tous ses MFrag originaux s'appliquent). Une distribution de probabilité peut être décrite en utilisant un format déclaratif dépendant de l'application, telles que la distribution de probabilité locale de UnBBayes (LPD), ou par le biais d'une table PR-OWL (qui a des assignations de probabilité dans ses cellules).

Il existe deux types de distributions de probabilité, la Table PR-OWL (*PROWLTable*) et la distribution déclarative (*DeclarativeDistribution*) **[51].**

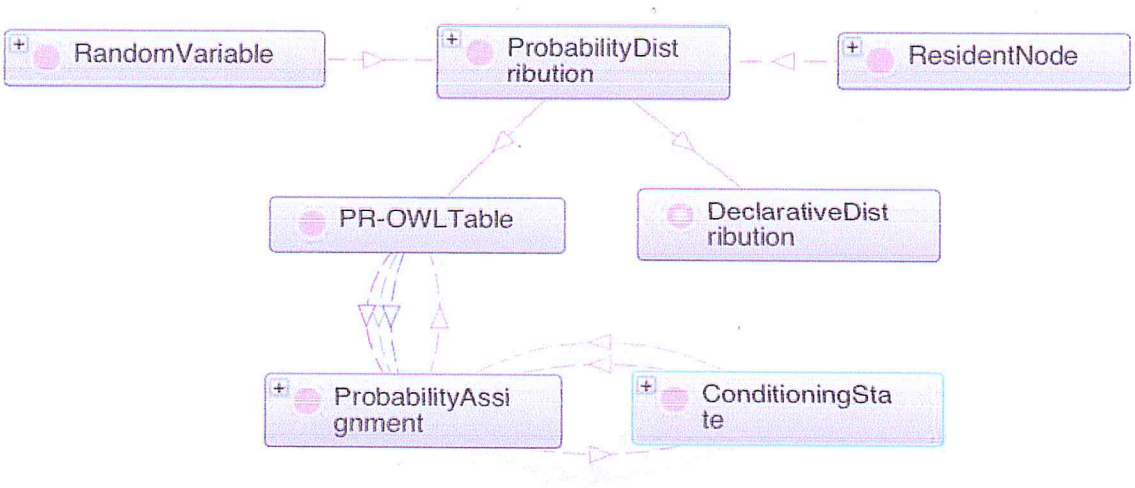

Figure 17 : Graphe avec les principaux concepts nécessaires pour définir une distribution de probabilités.

- **DeclarativeDistribution** : est définie par un script. C'est une distribution qui est transportée via le type de donnée *xsd:string* en utilisant un format spécifique défini dans la propriété de type de données (data type property) *hasDeclaration*. Afin de permettre un algorithme de MEBN de fonctionner, un analyseur devrait être capable d'extraire les informations de la distribution de probabilité dans le format qu'elle est stockée, puis transmettre cette information à l'algorithme MEBN dans son propre format spécifique à l'application.

  Cependant, on suppose que le raisonneur PR-OWL comprendrait le format dans lequel les informations sont stockées. Les tables de PR-OWL, d'autre part, transmettent les distributions de probabilités d'une manière plus interopérable, mais elles ne sont pas assez souples pour représenter les distributions complexes, tels que les cas dans lesquels un nœud a plusieurs parents possibles **[51].**

- **PR-OWLTable :** dispose toutes les assignations de probabilité pour chaque état d'une variable aléatoire ou nœud résident stocké dans un format *xsd:decimal*.

  Ce format de stockage des distributions de probabilité ne peut pas représenter les cas complexes pour lesquels seules les formules peuvent représenter une distribution de probabilité (par exemple un nœud qui a un nombre variable de parents), et peut être supporté habituellement par des grandes ontologies, puisque chaque table peut avoir de nombreuses cellules et chaque cellule est un individu de la classe *ProbabilityAssignment*. C'est pourquoi, les tables de PR-OWL ne sont recommandées

que pour les modèles les plus simples dans lesquels le niveau de compatibilité maximal est souhaité **[51]**.

⇨ **ProbabilityAssignment** :

Chaque cellule dans une table PR-OWL a une assignation de probabilité pour l'état d'une variable aléatoire ou d'un nœud résident étant donné les états de ses nœuds parents (les variables aléatoires ne disposent pas des nœuds parents). Ainsi, la relation résultante est n-aire, et elle est représentée via une propriété d'objet (object property) *hasProbabilityAssignment* qui comprend le nom de l'état dans lequel la probabilité est assignée (via la propriété de données (data property) *hasStateName*), la valeur de probabilité elle-même (via la propriété de données *hasStateProbability*), et la liste des états des nœuds parents (via la propriété d'objet *hasConditioningState*) qui définissent collectivement le contexte dans lequel cette assignation de probabilité est valide. En outre, les individus de la classe *ProbabilityttAssignment* ont la propriété d'objet *isProbabilityAssignmentIn* qui les relie avec leurs tables PR-OWL correspondante **[51]**.

⇨ **ConditioningState :**

En d'autres termes, il définit l'état d'un parent qui conditionne l'attribution de probabilité pour un état spécifique du nœud enfant. Les individus de cette classe sont utilisés pour construire la table de distribution probabiliste de PR-OWL. Chaque cellule d'une telle table correspond à une assignation de probabilité d'une valeur possible d'un nœud, étant donné une combinaison des états de ses parents, chaque individu de la classe *ConditioningState* représente un pair parent/état, donc une assignation de probabilité est conditionnée par un ensemble de pair *ConditioningState* **[51].**

⇨ **MExpression :**

Elle représente une formule ou un terme de la logique du premier ordre, qui a la variable aléatoire comme un élément principal. Le nombre d'arguments définis sur l'expression de MEBN doit être le même que le nombre d'arguments définis sur la variable aléatoire qu'il se réfère. En outre, les types d'arguments doivent être compatibles. Il existe essentiellement quatre types d'arguments qui peuvent être utilisés pour exprimer une expression de MEBN *ConstantArgument, OrdinaryVariableArgument, ExemplarArgument,* et *MExpressionArgument.*

Chapitre 4 : conception du système

1. Introduction :

La gestion de la connaissance incertaine est devenue un axe de recherche dans les dernières années, dans ce chapitre nous nous focalisons sur la modélisation de la connaissance incertaine dans les ontologies OWL, pour cela nous allons contribuer avec un système automatique qui est capable de transformer une ontologie classique formalisée en OWL en une ontologie probabiliste formalisée en PR-OWL, le tout en gardant la sémantique de la première ontologie.

2. Architecture globale de notre système :

L'objectif de notre travail est de transformer une ontologie OWL bien formalisée, qui comporte des informations incomplètes, en une ontologie probabiliste PR-OWL afin de supporter cette incomplétude.

L'incomplétude au niveau de l'ontologie OWL se trouve précisément dans la partie des instances de cette ontologie. Le diagramme de la figure suivante donne une vision générale de notre travail et les différentes étapes du processus de transformation sont illustrées dans le schéma suivant.

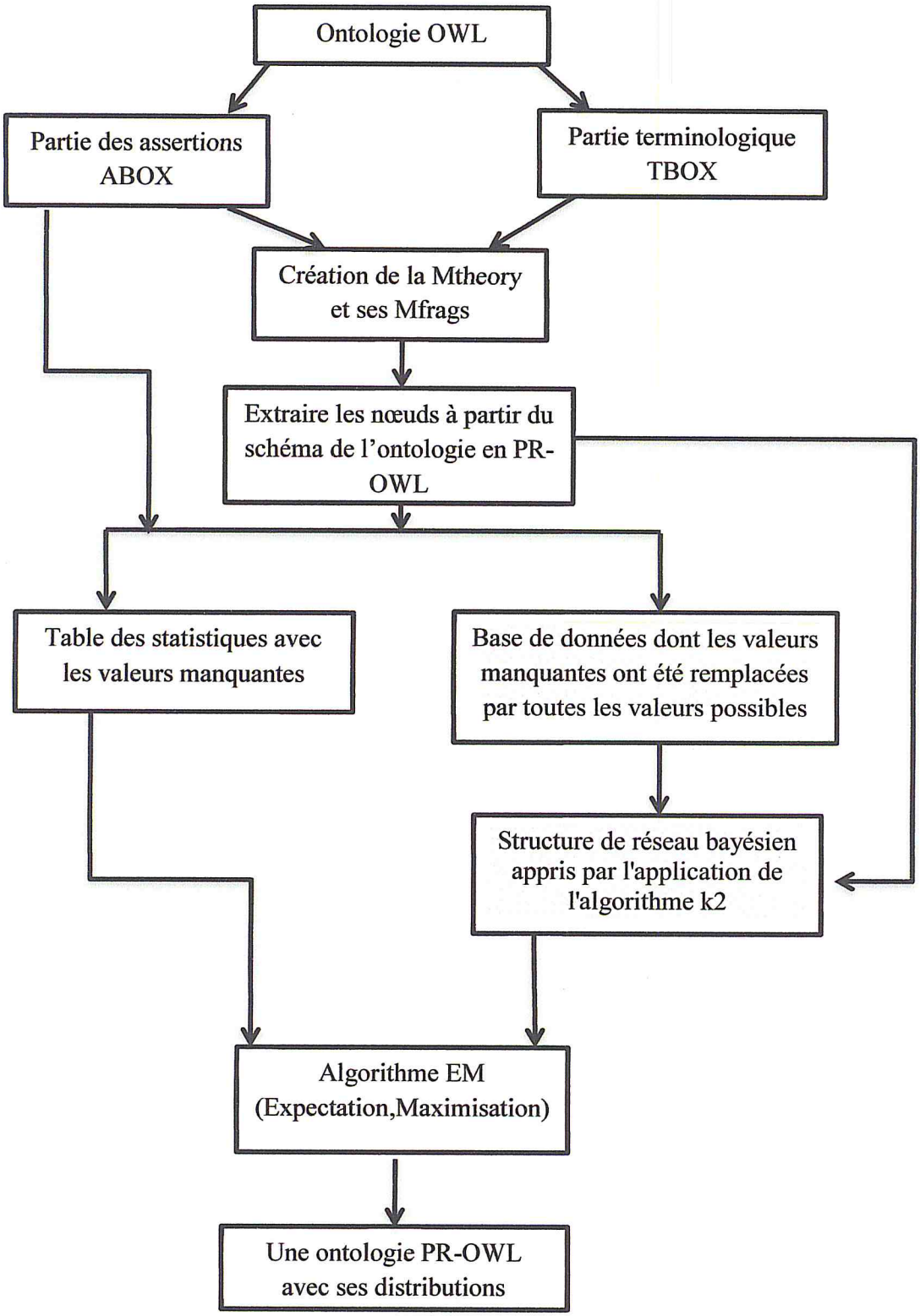

Figure 19 : système global

2.1 L'extraction des classes et des propriétés de l'ontologie :

2.1.1 L'extraction du schéma de l'ontologie Tbox :

Une ontologie OWL est caractérisée par sa partie TBOX ou ce qu'on appelle la partie schéma de l'ontologie qui va contenir :

- Les concepts.
- Les propriétés (relations).
- Le domaine et le range de chaque propriété.

Figure 20 : Exemple d'une hiérarchie des classes d'une ontologie

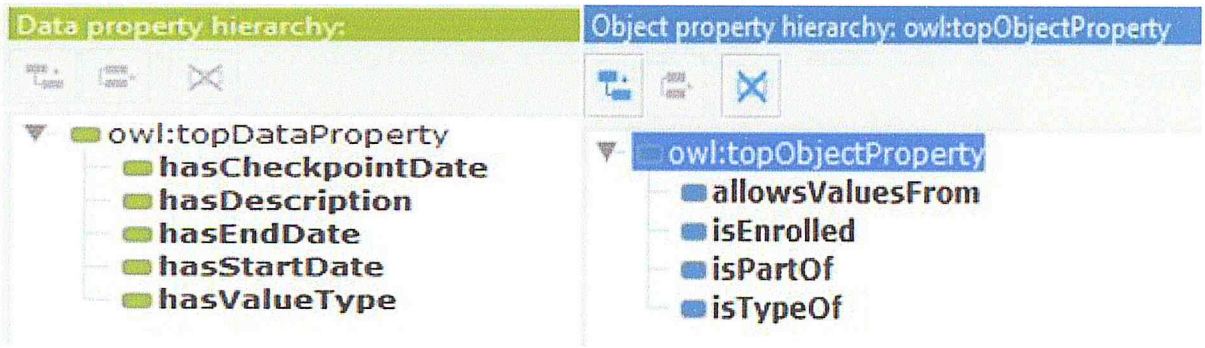

Figure 21 : Exemple des propriétés d'une ontologie (datatypeproperty et objectproperty)

2.1.2 L'extraction de la partie assertion de l'ontologie Abox :

Une ontologie OWL est caractérisée aussi par sa partie ABOX ou ce qu'on appelle la partie des assertions de l'ontologie, elle va contenir les Assertions des attributs.

Figure 22 : Exemple des individus d'une ontologie

Nous allons extraire ces parties pour pouvoir importer les classes, les propriétés, les relations entre les propriétés et leurs domaines et rangs, et les assertions qu'on aura besoin pour la construction de notre ontologie probabiliste, et pour l'application des algorithmes d'apprentissage EM et K2.

2.2 La création de l'ontologie probabiliste :

Apres l'extraction des TBoxs et ABoxs, la création de l'ontologie probabiliste se passe par les étapes suivantes :

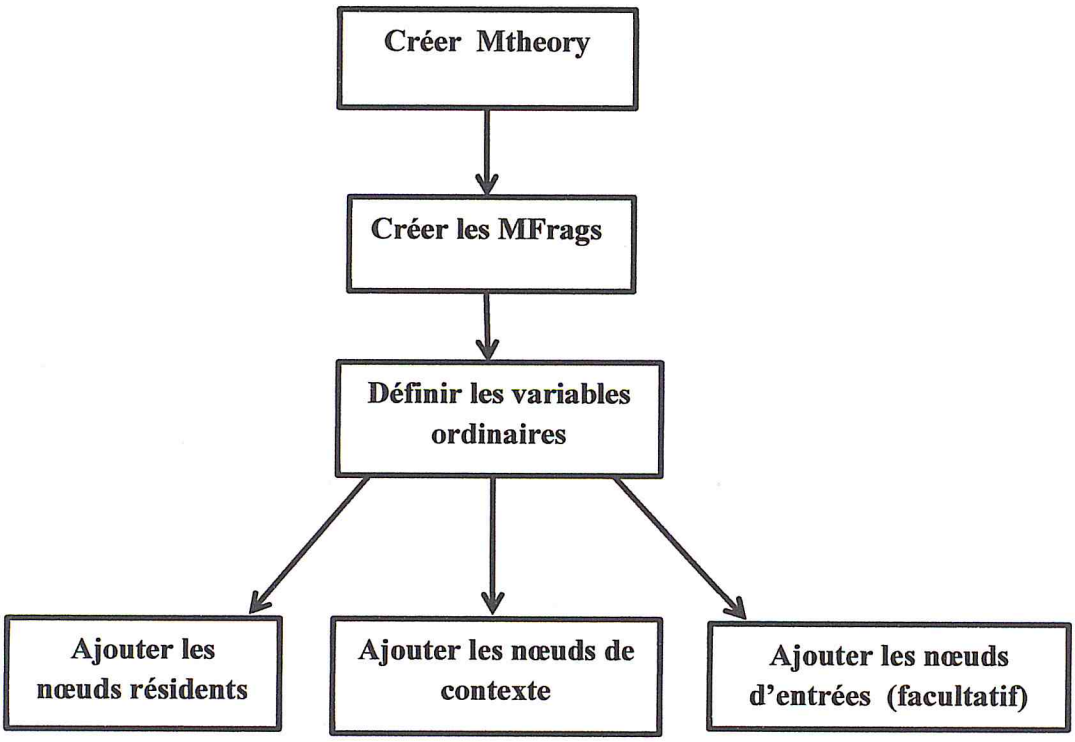

Figure 23 : étapes de la construction des concepts de l'ontologie probabiliste

2.2.1 Création de la MTheory :

La création de la Mtheory se fait juste après l'importation de l'ontologie qu'on veut modéliser, à notre application, cette MTheory va contenir les Mfrags qu'on aura besoin pour notre ontologie probabiliste.

La figure ci-dessous montre un exemple de la syntaxe PR-OWL correspondante à la création d'une MTheory

```
414    <ClassAssertion>
415        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MTheory"/>
416        <NamedIndividual IRI="#FraudPreventionAndDetectionInPublicProcurements_MTheory"/>
417    </ClassAssertion>
```

Figure 24 : Exemple de création d'une MTheory

2.2.2 MFrag :

Une fois que nous avons créé notre MTheory, nous commençons par la création des MFrags qui vont contenir les nœuds (nœuds résidents, nœuds de contexte, nœuds d'entrés) qui sont les variables aléatoires, pour notre cas nous utilisons que les MFrags de domaine du moment qu'on traite une ontologie spécifique à un domaine.

La figure ci-dessous montre un exemple de la syntaxe PR-OWL correspondante à la création d'un MFrag de domaine.

```
254    <ClassAssertion>
255        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#DomainMFrag"/>
256        <NamedIndividual IRI="#Domain_MFrag.EnterpriseInfo_MFrag"/>
257    </ClassAssertion>
2380   <ObjectPropertyAssertion>
2381       <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMFragOf"/>
2382       <NamedIndividual IRI="#Domain_MFrag.EnterpriseInfo_MFrag"/>
2383       <NamedIndividual IRI="#FraudPreventionAndDetectionInPublicProcurements_MTheory"/>
2384   </ObjectPropertyAssertion>
3400   <ObjectPropertyAssertion>
3401       <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasMFrag"/>
3402       <NamedIndividual IRI="#FraudPreventionAndDetectionInPublicProcurements_MTheory"/>
3403       <NamedIndividual IRI="#Domain_MFrag.EnterpriseInfo_MFrag"/>
3404   </ObjectPropertyAssertion>
```

Figure 25 : Exemple de création d'un MFrag de domaine

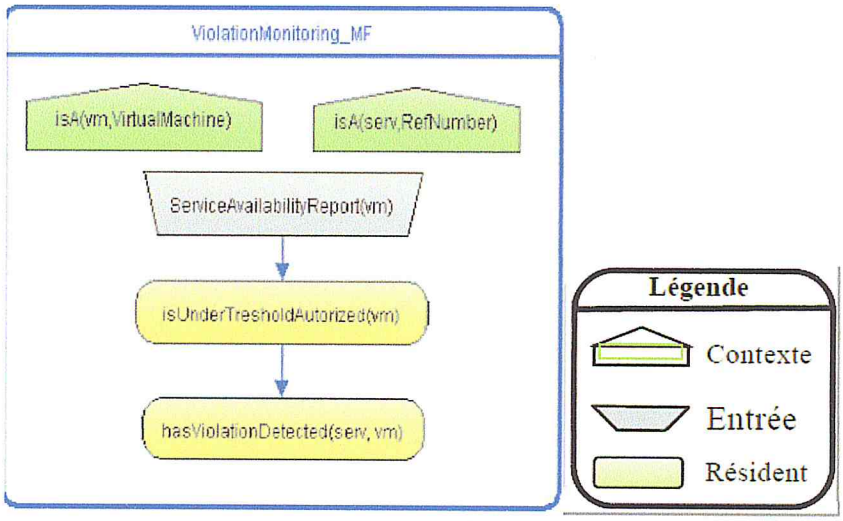

Figure 26: exemple d'un MFrag complet

2.2.2.1 Déclaration les variables ordinaires :

Pour la déclaration des variables ordinaires on aura besoin des classes OWL que nous avons déjà extrait. Une variable ordinaire est déclarée avec la relation 'est-un' (IsA), c'est une instanciation d'une entité (classe OWl). Par exemple : IsA(p, personne) tels que p est notre variable ordinaire et personne est l'entité (la classe OWL).

La figure ci-dessous montre un exemple de la syntaxe PR-OWL correspondante à la création d'une variable ordinaire :

```
2340      <ObjectPropertyAssertion>
2341          <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasOrdinaryVariable"/>
2342          <NamedIndividual IRI="#Domain_MFrag.CompetitionCompromised_MFrag"/>
2343          <NamedIndividual IRI="#CompetitionCompromised_MFrag.enterprise"/>
2344      </ObjectPropertyAssertion>
238       <ClassAssertion>
239           <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariable"/>
240           <NamedIndividual IRI="#CompetitionCompromised_MFrag.enterprise"/>
241       </ClassAssertion>
6810  <DataPropertyAssertion>
6811      <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isSubstitutedBy"/>
6812      <NamedIndividual IRI="#CompetitionCompromised_MFrag.enterprise"/>
6813      <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#anyURI">file:/E:/MEBN/Ontologias/CGU/FraudInPublicProcurement.owl#Enterprise</Literal>
6814  </DataPropertyAssertion>
8192      <AnnotationAssertion>
8193          <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
8194          <IRI>#CompetitionCompromised_MFrag.enterprise</IRI>
8195          <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">OX3</Literal>
8196      </AnnotationAssertion>
```

Figure 27 : Exemple de la création d'une variable ordinaire

2.2.2.2 ResidentNode (nœud résident) :

Apres la création d'un MFrags, on doit définir au moins un nœud résident dans ce MFrag,

La figure ci-dessous montre les étapes de la création d'un nœud résident :

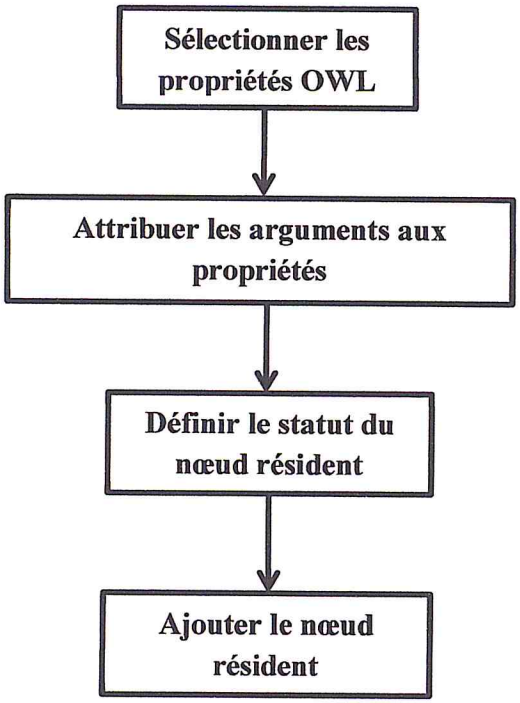

Figure 28 : étapes de la création d'un nœud résident

- **Sélectionner les propriétés OWL:**

  Pour définir un nœud résident, nous devons choisir quelle est la propriété OWL que nous devons utiliser, cette propriété sera par la suite le nœud résident mais en lui donnant leurs arguments ainsi en définissant son état (variable de sortie du nœud).

- **Attribuer les arguments aux propriétés :**

  Les arguments sont les variables ordinaires définies précédemment, un nœud résident peut avoir de un à deux arguments.

- **Définir le statut du nœud résident:**

  La définition de l'état du nœud a pour objectif de donner le type de valeurs possibles du nœud, elles peuvent être booléennes ou des instances de l'ontologie OWL (prédéfinies).

- **Ajouter le nœud résident:**

  Apres l'exécution de toutes les taches précédente on pourra ajouter le nœud résident.

  La figure ci-dessous montre la syntaxe PR-OWL correspondante à la création d'un nœud résident :

```
1166    <ClassAssertion>
1167        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#BooleanRandomVariable"/>
1168        <NamedIndividual IRI="#RV_existsFrontInEnterprise"/>
1169    </ClassAssertion>
207     <AnnotationAssertion>
208         <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
209         <IRI>#Domain_MFrag.EnterpriseInfo_MFrag</IRI>
210         <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string"></Literal>
211     </AnnotationAssertion>
8401    <AnnotationAssertion>
8402        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
8403        <IRI>#ExistsFrontInEnterprise_MFrag.enterprise</IRI>
8404        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">OX2</Literal>
8405    </AnnotationAssertion>
8406    <AnnotationAssertion>
8407        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
8408        <IRI>#ExistsFrontInEnterprise_MFrag.person</IRI>
8409        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">OX1</Literal>
8410    </AnnotationAssertion>
1714    <ClassAssertion>
1715        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#DeclarativeDistribution"/>
1716        <NamedIndividual IRI="http://www.pr-owl.org/pr-owl2.owl#existsFrontInEnterprise_Table"/>
1717    </ClassAssertion>
302     <ClassAssertion>
303         <Class IRI="http://www.pr-owl.org/pr-owl2.owl#DomainResidentNode"/>
304         <NamedIndividual IRI="#Domain_Res.existsFrontInEnterprise"/>
305     </ClassAssertion>
7190    <DataPropertyAssertion>
7191        <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgumentNumber"/>
7192        <NamedIndividual IRI="#RV_existsFrontInEnterprise_1"/>
7193        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">1</Literal>
7194    </DataPropertyAssertion>
4870    <ObjectPropertyAssertion>
4871        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
4872        <NamedIndividual IRI="#MEXPRESSION_existsFrontInEnterprise"/>
4873        <NamedIndividual IRI="#existsFrontInEnterprise_1"/>
4874    </ObjectPropertyAssertion>
4875    <ObjectPropertyAssertion>
4876        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMExpressionOf"/>
4877        <NamedIndividual IRI="#MEXPRESSION_existsFrontInEnterprise"/>
4878        <NamedIndividual IRI="#Domain_Res.existsFrontInEnterprise"/>
4879    </ObjectPropertyAssertion>
4880    <ObjectPropertyAssertion>
4881        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfMExpression"/>
4882        <NamedIndividual IRI="#MEXPRESSION_existsFrontInEnterprise"/>
4883        <NamedIndividual IRI="#RV_existsFrontInEnterprise"/>
4884    </ObjectPropertyAssertion>
2880    <ObjectPropertyAssertion>
2881        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasMExpression"/>
2882        <NamedIndividual IRI="#Domain_Res.existsFrontInEnterprise"/>
2883        <NamedIndividual IRI="#MEXPRESSION_existsFrontInEnterprise"/>
2884    </ObjectPropertyAssertion>
2405    <ObjectPropertyAssertion>
2406        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasResidentNode"/>
2407        <NamedIndividual IRI="#Domain_MFrag.ExistsFrontInEnterprise_MFrag"/>
2408        <NamedIndividual IRI="#Domain_Res.existsFrontInEnterprise"/>
2409    </ObjectPropertyAssertion>
```

```
    <DataPropertyAssertion>
        <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgumentNumber"/>
        <NamedIndividual IRI="#existsFrontInEnterprise_1"/>
        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">1</Literal>
    </DataPropertyAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
        <NamedIndividual IRI="#existsFrontInEnterprise_1"/>
        <NamedIndividual IRI="#MEXPRESSION_existsFrontInEnterprise"/>
    </ObjectPropertyAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isResidentNodeIn"/>
        <NamedIndividual IRI="#Domain_Res.existsFrontInEnterprise"/>
        <NamedIndividual IRI="#Domain_MFrag.ExistsFrontInEnterprise_MFrag"/>
    </ObjectPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isSubstitutedBy"/>
    <NamedIndividual IRI="#EnterpriseInfo_MFrag.enterprise"/>
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#anyURI">file:/E:/MEBN/Ontologias/CGU/FraudInPublicProcurement.owl#Enterprise</Literal>
</DataPropertyAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
        <NamedIndividual IRI="#RV_existsFrontInEnterprise"/>
        <NamedIndividual IRI="#RV_existsFrontInEnterprise_1"/>
    </ObjectPropertyAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfMExpression"/>
        <NamedIndividual IRI="#RV_existsFrontInEnterprise"/>
        <NamedIndividual IRI="#MEXPRESSION_existsFrontInEnterprise"/>
    </ObjectPropertyAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfArgumentIn"/>
        <NamedIndividual IRI="#ProcurementInfo_MFrag.enterprise"/>
        <NamedIndividual IRI="#isSuspended_1"/>
    </ObjectPropertyAssertion>
    <ClassAssertion>
        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MappingArgument"/>
        <NamedIndividual IRI="#RV_existsFrontInEnterprise_1"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariableArgument"/>
        <NamedIndividual IRI="#existsFrontInEnterprise_1"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#RandomVariable"/>
        <NamedIndividual IRI="#RV_existsFrontInEnterprise"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#DomainResidentNode"/>
        <NamedIndividual IRI="#Domain_Res.existsFrontInEnterprise"/>
    </ClassAssertion>
    <ClassAssertion>
        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#SimpleMExpression"/>
        <NamedIndividual IRI="#MEXPRESSION_existsFrontInEnterprise"/>
    </ClassAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfArgument"/>
        <NamedIndividual IRI="#MEXPRESSION_CX10_1"/>
        <NamedIndividual IRI="#ProcurementInfo_MFrag.enterprise"/>
    </ObjectPropertyAssertion>
```

Figure 29 : Exemple de la création d'un nœud résident

2.2.2.3 ContextNode (nœud de contexte) :

Les nœuds de contextes définissent les contraintes sur le MFrag là où ils sont définis.

L'ajout d'un nœud de contexte passe par les étapes suivantes :

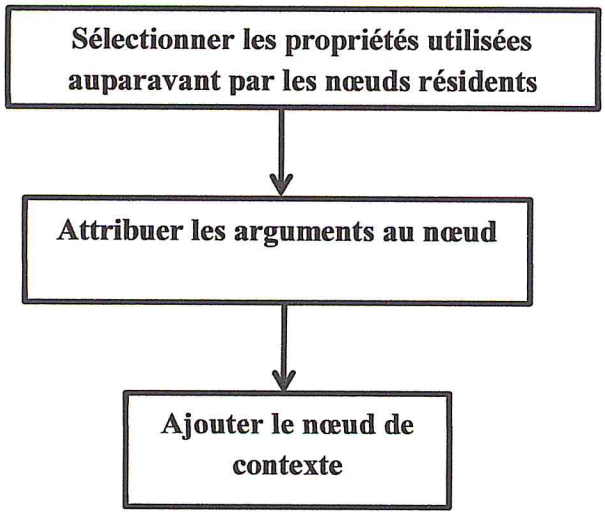

Figure 30 : Etapes de la création d'un nœud de contexte

- **Sélectionner les propriétés utilisées auparavant par les nœuds résidents:**
  Pour notre cas, le nœud de contexte se crée par une propriété qui a était déjà utilisée pour les nœuds résidents, il y'a plusieurs cas de nœuds contexte, des cas là où on aura besoin de donner des contraintes aux variables ordinaires (les quantificateurs, les opérateurs logiques).
  Ce nœud de contexte sera la contrainte sur le MFrag là où il est défini, elle doit être satisfaite pour que le MFrag soit valide.
- **Attribuer les arguments au nœud choisi :**
  Nous devons attribuer les arguments (qui sont les variables ordinaires) au nœud, elles doivent être déclarées dans le même MFrag là où le nœud de contexte est défini.
- **Ajouter le nœud de contexte :**
  Apres l'exécution des taches précédentes on pourra ajouter un nœud de contexte. La figure suivante montre la syntaxe PR-OWL correspondante à la création d'un nœud de contexte :

```
626     <ClassAssertion>
627         <Class IRI="http://www.pr-owl.org/pr-owl2.owl#BooleanMExpression"/>
628         <NamedIndividual IRI="#MEXPRESSION_CX10"/>
629     </ClassAssertion>
630     <ClassAssertion>
631         <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MExpression"/>
632         <NamedIndividual IRI="#MEXPRESSION_CX10"/>
633     </ClassAssertion>
8077    <AnnotationAssertion>
8078        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
8079        <IRI>#CX10</IRI>
8080        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">CX10</Literal>
8081    </AnnotationAssertion>
320     <ObjectPropertyAssertion>
321         <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasContextNode"/>
322         <NamedIndividual IRI="#Domain_MFrag.CompetitionCompromised_MFrag"/>
323         <NamedIndividual IRI="#CX10"/>
324     </ObjectPropertyAssertion>

    <ObjectPropertyAssertion>
        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasMExpression"/>
        <NamedIndividual IRI="#CX10"/>
        <NamedIndividual IRI="#MEXPRESSION_CX10"/>
    </ObjectPropertyAssertion>
34      <ClassAssertion>
35          <Class IRI="http://www.pr-owl.org/pr-owl2.owl#ContextNode"/>
36          <NamedIndividual IRI="#CX10"/>
37      </ClassAssertion>
3825    <ObjectPropertyAssertion>
3826        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
3827        <NamedIndividual IRI="#MEXPRESSION_CX10"/>
3828        <NamedIndividual IRI="#MEXPRESSION_CX10_2"/>
3829    </ObjectPropertyAssertion>
3830    <ObjectPropertyAssertion>
3831        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
3832        <NamedIndividual IRI="#MEXPRESSION_CX10"/>
3833        <NamedIndividual IRI="#MEXPRESSION_CX10_1"/>
3834    </ObjectPropertyAssertion>
3835    <ObjectPropertyAssertion>
3836        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMExpressionOf"/>
3837        <NamedIndividual IRI="#MEXPRESSION_CX10"/>
3838        <NamedIndividual IRI="#CX10"/>
3839    </ObjectPropertyAssertion>
3840    <ObjectPropertyAssertion>
3841        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfMExpression"/>
3842        <NamedIndividual IRI="#MEXPRESSION_CX10"/>
3843        <NamedIndividual IRI="#RV_isParticipantIn"/>
3844    </ObjectPropertyAssertion>
3845    <ObjectPropertyAssertion>
3846        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
3847        <NamedIndividual IRI="#MEXPRESSION_CX10_1"/>
3848        <NamedIndividual IRI="#MEXPRESSION_CX10"/>
3849    </ObjectPropertyAssertion>
3850    <ObjectPropertyAssertion>
3851        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfArgument"/>
3852        <NamedIndividual IRI="#MEXPRESSION_CX10_1"/>
3853        <NamedIndividual IRI="#ProcurementInfo_MFrag.enterprise"/>
3854    </ObjectPropertyAssertion>
```

- **Sélectionner le nœud résident avec ses arguments :**

  On doit choisir parmi les nœuds résidents déjà définis dans les autres MFrags le nœud d'entrée qui va influencer sur quelques nœuds résidents qui sont dans le même MFrag que ce nœud d'entrée.

- **Ajouter le nœud d'entrée :**

  Apres la sélection du nœud résident, on pourra ajouter un nœud d'entrée dans le MFrag.

La figure suivante montre la syntaxe PR-OWL correspondante à la création d'un nœud d'entrée :

```
8434    <AnnotationAssertion>
8435        <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
8436        <IRI>#IX1</IRI>
8437        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">IX1</Literal>
8438    </AnnotationAssertion>
470     <ClassAssertion>
471         <Class IRI="http://www.pr-owl.org/pr-owl2.owl#GenerativeInputNode"/>
472         <NamedIndividual IRI="#IX1"/>
473     </ClassAssertion>
    <ObjectPropertyAssertion>
        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
        <NamedIndividual IRI="#MEXPRESSION_IX1"/>
        <NamedIndividual IRI="#IX1_1"/>
    </ObjectPropertyAssertion>
6910    <DataPropertyAssertion>
6911        <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgumentNumber"/>
6912        <NamedIndividual IRI="#IX1_1"/>
6913        <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#integer">1</Literal>
6914    </DataPropertyAssertion>
!430    <ObjectPropertyAssertion>
!431        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasInputNode"/>
!432        <NamedIndividual IRI="#Domain_MFrag.FrontOfEnterprise_MFrag"/>
!433        <NamedIndividual IRI="#IX1"/>
!434    </ObjectPropertyAssertion>
0   <ObjectPropertyAssertion>
1       <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasInputInstance"/>
2       <NamedIndividual IRI="#Domain_Res.hasValue"/>
3       <NamedIndividual IRI="#IX1"/>
4   </ObjectPropertyAssertion>
!465    <ObjectPropertyAssertion>
!466        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasMExpression"/>
!467        <NamedIndividual IRI="#IX1"/>
!468        <NamedIndividual IRI="#MEXPRESSION_IX1"/>
!469    </ObjectPropertyAssertion>
3470    <ObjectPropertyAssertion>
3471        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isInputNodeIn"/>
3472        <NamedIndividual IRI="#IX1"/>
3473        <NamedIndividual IRI="#Domain_MFrag.FrontOfEnterprise_MFrag"/>
3474    </ObjectPropertyAssertion>
```

```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isContextNodeIn"/>
    <NamedIndividual IRI="#CX10"/>
    <NamedIndividual IRI="#Domain_MFrag.CompetitionCompromised_MFrag"/>
</ObjectPropertyAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#BooleanMExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_CX10"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_CX10"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariableArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX10_1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariableArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX10_2"/>
</ClassAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
    <NamedIndividual IRI="#MEXPRESSION_CX10_2"/>
    <NamedIndividual IRI="#MEXPRESSION_CX10"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX10_2"/>
    <NamedIndividual IRI="#ProcurementInfo_MFrag.procurement"/>
</ObjectPropertyAssertion>
```

Figure 31 : Exemple de la création d'un nœud de contexte

2.2.2.4 InputNode (nœud d'entrée) :

C'est une variable aléatoire qui a sa distribution définie dans un autre MFrag, mais sa valeur influence sur quelques nœuds résidents dans le MFrag là où on ajoute le nœud d'entrée.

L'ajout d'un nœud d'entrée se passe par les étapes suivantes :

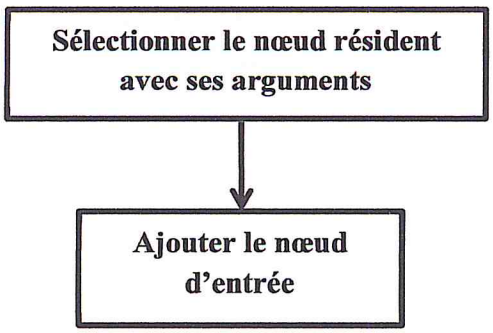

Figure 32 : Etapes de la création d'un nœud d'entrée

```
3665    <ObjectPropertyAssertion>
3666        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
3667        <NamedIndividual IRI="#IX1_1"/>
3668        <NamedIndividual IRI="#MEXPRESSION_IX1"/>
3669    </ObjectPropertyAssertion>
515     <ObjectPropertyAssertion>
516         <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMExpressionOf"/>
517         <NamedIndividual IRI="#MEXPRESSION_IX1"/>
518         <NamedIndividual IRI="#IX1"/>
519     </ObjectPropertyAssertion>
525     <ObjectPropertyAssertion>
526         <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfArgumentIn"/>
527         <NamedIndividual IRI="#ProcurementInfo_MFrag.procurement"/>
528         <NamedIndividual IRI="#IX1_1"/>
529     </ObjectPropertyAssertion>
5775    <ObjectPropertyAssertion>
5776        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfMExpression"/>
5777        <NamedIndividual IRI="#RV_hasValue"/>
5778        <NamedIndividual IRI="#MEXPRESSION_IX1"/>
5779    </ObjectPropertyAssertion>
950     <ClassAssertion>
951         <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MExpression"/>
952         <NamedIndividual IRI="#MEXPRESSION_IX1"/>
953     </ClassAssertion>
    <ClassAssertion>
        <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariableArgument"/>
        <NamedIndividual IRI="#IX1_1"/>
    </ClassAssertion>
3670    <ObjectPropertyAssertion>
3671        <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfArgument"/>
3672        <NamedIndividual IRI="#IX1_1"/>
3673        <NamedIndividual IRI="#ProcurementInfo_MFrag.procurement"/>
3674    </ObjectPropertyAssertion>
620     <ObjectPropertyAssertion>
621         <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfMExpression"/>
622         <NamedIndividual IRI="#MEXPRESSION_IX1"/>
623         <NamedIndividual IRI="#RV_hasValue"/>
624     </ObjectPropertyAssertion>
```

Figure 33 : Exemple de la création d'un nœud d'entrée

À la fin de ces traitements on obtiendra une ontologie probabiliste PR-OWL qui contient les classes et les propriétés nécessaires pour passer à l'étape suivante qui sera la construction du réseau bayésien.

2.3 Construction d'une structure initiale :

Pour construire la structure globale de réseau bayésien à partir de la spécification des MFrags on doit réunir tous les sous-structure qui se trouvent dans les MFrags, et ceci est fait par l'utilisation des nœuds d'entrés (Input nouds) pour faire le lien entre les sous-structures. Cependant, la structure obtenu elle n'est pas toujours optimale pour cela on doit faire appel à un algorithme d'apprentissage de structure pour améliorer la qualité du structure de réseau bayésien.

Nous allons appliquer l'algorithme K2 pour l'apprentissage de la structure, ce dernier a besoin comme paramètres une base de données (dans notre cas nous allons utiliser la partie des assertions comme une source de données), ainsi un ordre de causalité des nouds (nous allons construire cet ordre à partir de la structure de la spécification)

2.3.1 Types des nœuds du réseau bayésien :

Le type de relation entre les nouds qui vont être inclus dans la structure du réseau est de type causal, on distingue deux types de nœud dans une relation de causalité :

- Relation de causalité entre les nœuds d'entrées avec des nœuds résidents. La figure ci-dessous montre un exemple de ce genre de relation :

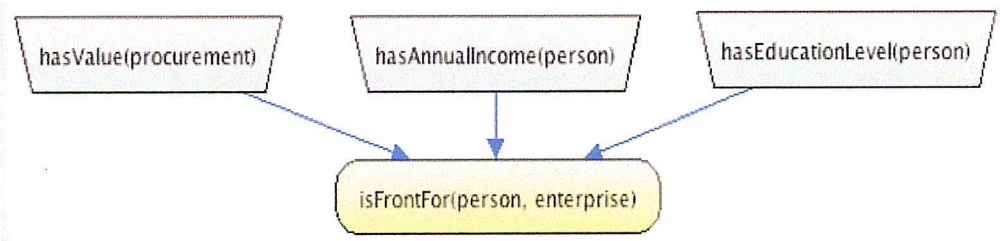

Figure 34 : exemple d'une relation de causalité entre les nœuds d'entrés et les nouds résidents

Tel que les nœuds représentés dans des trapèzes en gris sont des nouds d'entrées, et le nœud représenté dans un rectangle à coin arrondis en beige est un nœud résident.

- Relation de causalité entre les nœuds résidents avec d'autres nœuds résidents. La figure ci-dessous montre un exemple de ce genre de relation :

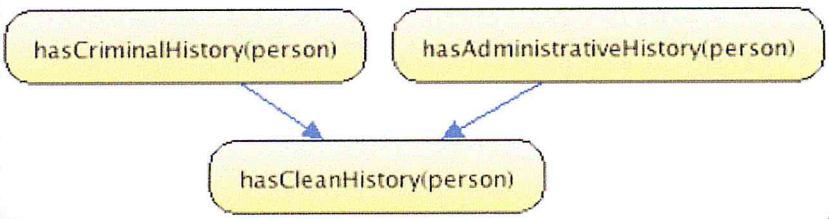

Figure 35: exemple d'une relation de causalité entre des nœuds résidents avec d'autres nœuds résidents

Tel que les nœuds représentés dans des rectangles à coin arrondis en beige sont des nœuds résidents.

2.3.2 Étapes pour faire la liaison entre nœuds :

Nous allons extraire de notre schéma PR-OWL les types de causalité citée précédemment. L'extraction passe par les étapes suivantes:

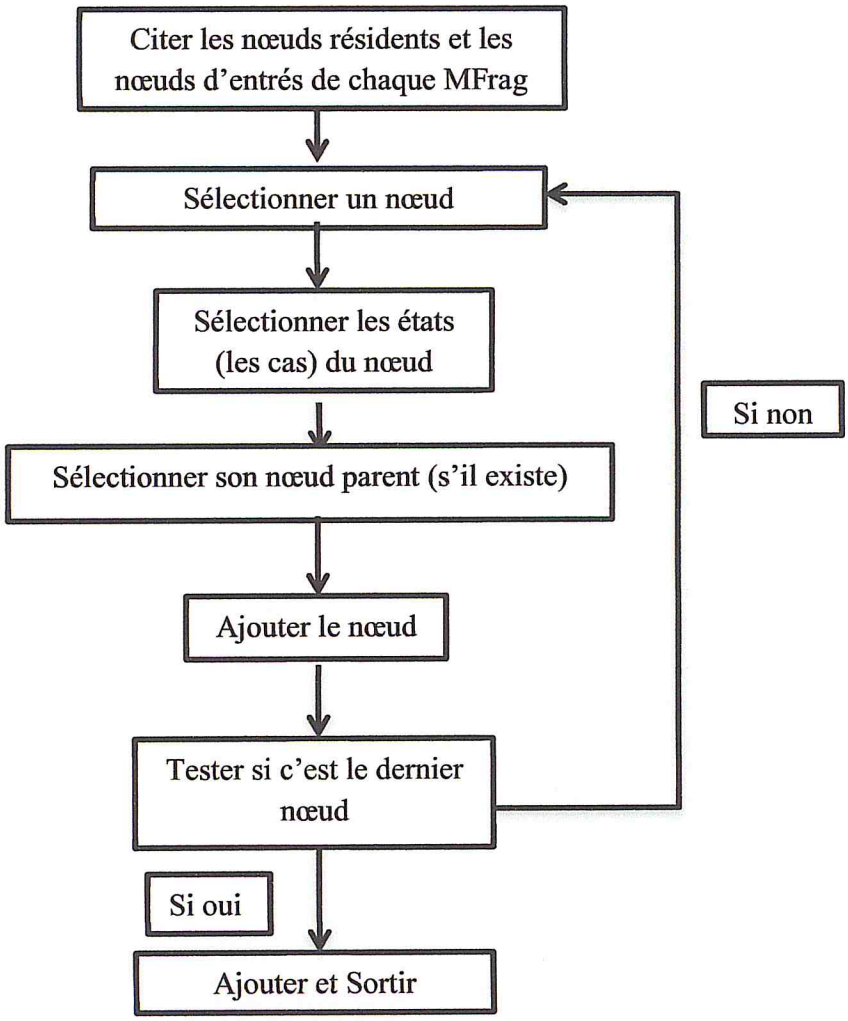

Figure 36 : étapes de la construction de la structure du réseau bayésien

- **Sélectionner un nœud** : nous sélectionnons un nœud parmi les nœuds extraits.
- **Sélectionner les états (les cas) du nœud :** après le choix du nœud, on sélectionne ses états qui sont aussi les états de ce nœud (résident) dans l'ontologie probabiliste PR-OWL.
- **Sélectionner son nœud parent (s'il existe) :** nous choisissons le parent de chaque nœud (s'il existe) selon la structure des MFrags de l'ontologie PR-OWL.
- **Ajouter le nœud :** après l'effectuation des étapes précédentes nous pouvons ajouter le nœud.

- **Tester si c'est le dernier nœud :** nous devons tester après chaque itération si on est arrivé au dernier nœud, si c'est le cas on enregistre la structure et on sort si non, on refait les étapes comme montré dans la figure.

2.4 Base de données des valeurs:

L'algorithme K2 a besoin d'une base de données pour générer le nouveau réseau, pour cela nous allons construire une base de donnée qui va contenir les informations nécessaires pour la construction du réseau en créant une table de statistiques.

La structure de la table est montrée dans le tableau suivant :

Nœud 1	Nœud..	Nœud n

Tableau 2 : la structure de la table des statistiques

Tel que les colonnes nommées par 'nœud' représentent les nœuds résidents de l'ontologie PR-OWL, les lignes représentent les valeurs des nœuds qui sont les états des nœuds qui sont les valeurs des ranges de la propriété correspondante, dans le cas de l'absence d'une valeur, les valeurs manquantes vont être remplacées par toutes les valeurs possibles. Ces états sont représentés par des valeurs numériques données par ordre ascendant.

L'exemple suivant montre un exemple d'une base de données :

```
@relation test

@attribute x {0,1}
@attribute y {0,1,2}
@attribute z {0,1}

@data
0,1,0
1,0,1
1,1,1
1,2,1
0,0,0
```

Figure 37 : exemple d'une base de données

Tel que X, Y, Z sont les nœuds, et les chiffres représentent les états des nœuds.

Les étapes de la construction de la base de données sont les suivantes :

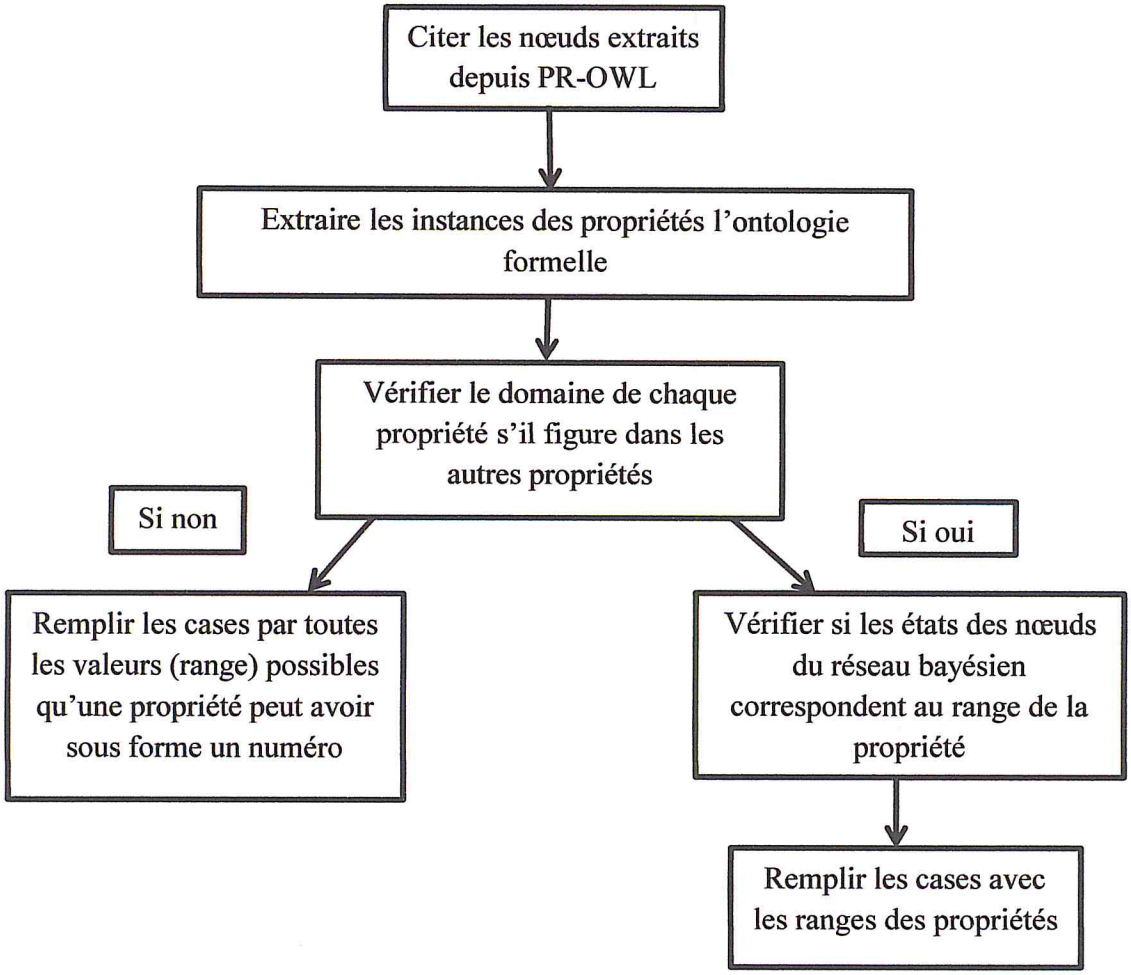

Figure 38 : Etapes de la création de la base de données pour l'algorithme K2

- **Citer les nœuds extraits depuis PR-OWL :** pour qu'on puisse faire les comparaisons nous devons d'abord citer tous les nœuds résidents et les nœuds d'entrés de la MTheory.
- **Extraire les relations entre les propriétés et les instances de l'ontologie formelle :** nous extrairions les propriétés (qui correspondent ou mappent les nœuds résidents) avec leurs domaine et range.
- **Vérifier le domaine de chaque propriété s'il figure dans les autres propriétés :** nous vérifions les domaines d'une propriété quelconque s'ils existent dans les autres propriétés qui font partie des nœuds de la MTheory.

- **Vérifier si les états des nœuds extraits correspondent aux ranges de la propriété :** Si le domaine d'une propriété quelconque figure dans les autres propriétés, nous vérifions si les ranges de ces propriétés correspondent aux états définis auparavant dans le nœud extrait (qui lui-même représente une propriété).
- **Remplir les cases par toutes les valeurs (range) possibles qu'une propriété peut avoir sous forme un numéro:** dans le cas où le domaine d'une propriété ne figure pas dans les autres propriétés, nous remplissons leurs cases par toutes les valeurs possible que les ranges de cette propriété peuvent avoir.
- **Remplir les cases avec les ranges des propriétés :** si le range d'une propriété quelconque correspond à l'état du nœud, nous remplissons la case avec la valeur du range.

2.5 La construction de la table des statistiques :

Nous utilisons la table de statistique pour faire l'apprentissage des paramètres, c.à.d. pour remplir les tables de probabilité, pour ce faire on applique l'algorithme EM, où nous considérons les statistiques sont incomplètes vu que les instances de l'ontologie sont incomplètes. Nous avons utilisé les instances de l'ontologie OWL, comme des statistiques manquantes, pour appliquer l'algorithme EM.

Apres l'extraction des nœuds du PR-OWL, la construction de la table des statistiques passe par les étapes suivantes :

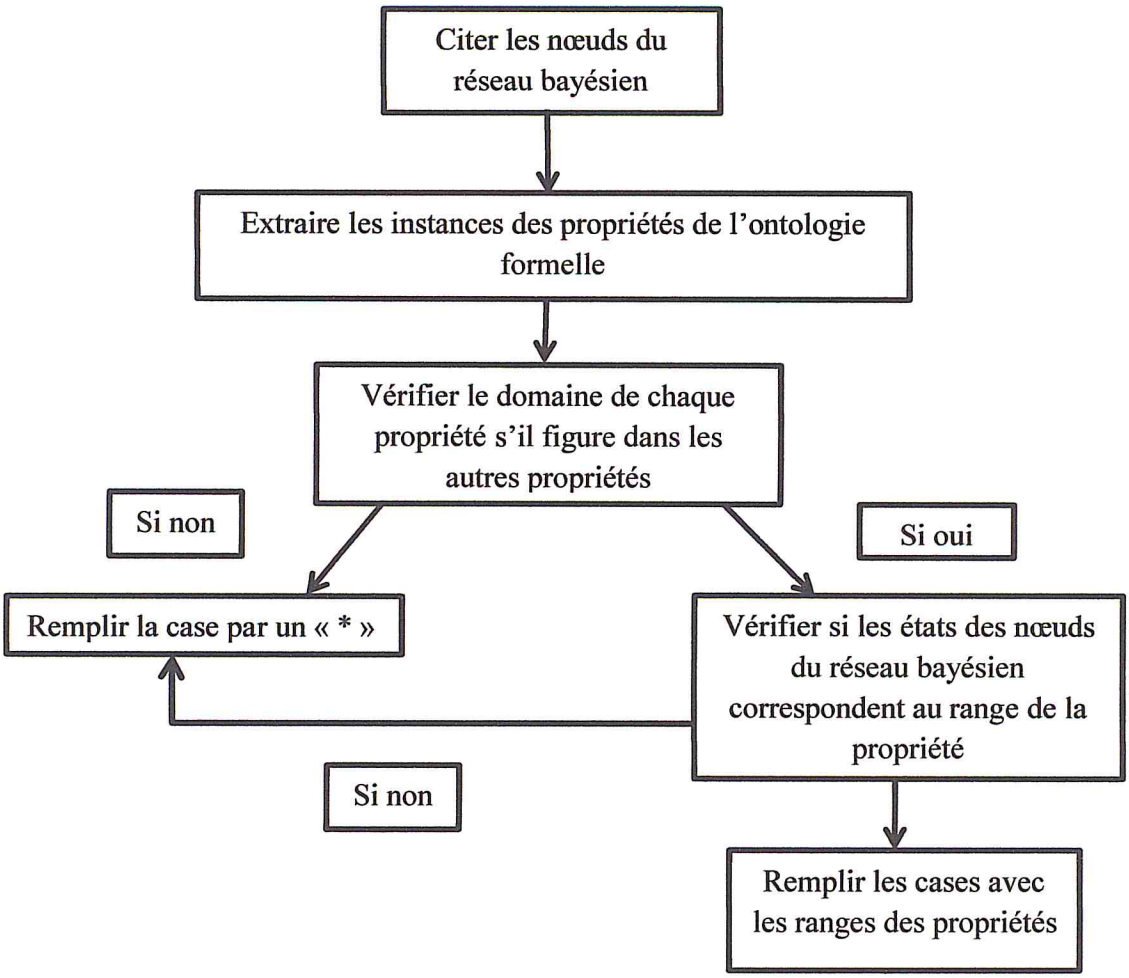

Figure 39 : Etapes de la construction de la table des statistiques avec des données manquantes

La construction de la table des statistiques avec des données manquantes passe par les mêmes étapes que la construction de la base de données de l'algorithme K2, sauf qu'à la place de remplir les cases des nœuds dont lequel le domaine d'une propriété ne figure pas dans les autres propriétés ou le cas où aucun état ne correspond à la valeur du range de la propriété :

- **Remplir la case par un « * » :** nous remplissons leurs cases par une étoile « * ».

	IDnum	R	S	T	A
1					
2	1	*	*	*	false
3	2	*	false	false	*
4	3	*	*	*	true
5	4	*	*	true	false
6	5	true	false	*	*
7	6	*	*	*	true
8	7	*	*	true	*
9	8	false	*	false	false
10	9	false	*	false	false
11	10	*	true	false	false
12	11	false	false	true	*
13	12	false	*	*	false
14	13	*	*	*	false
15	14	*	false	*	*
16	15	true	*	*	*
17	16	*	*	false	false
18	17	false	false	*	*
19	18	*	true	false	*
20	19	true	*	false	true

Figure 40 : Exemple d'une table de statistiques

Le but de la table des statistiques est de pouvoir utiliser l'algorithme EM pour l'apprentissage des paramètres manquants. La table des statistiques sera par la suite un paramètre de d'entrée de l'algorithme EM.

2.6 Structure du réseau bayésien appris par l'application de l'algorithme k2 :

Après avoir créée la base de données et après l'extraction des nœuds depuis PR-OWL, l'apprentissage de la structure permet de trouver la meilleure structure du réseau.

Nous allons utiliser l'algorithme K2 pour optimiser la structure du réseau bayésien construit.

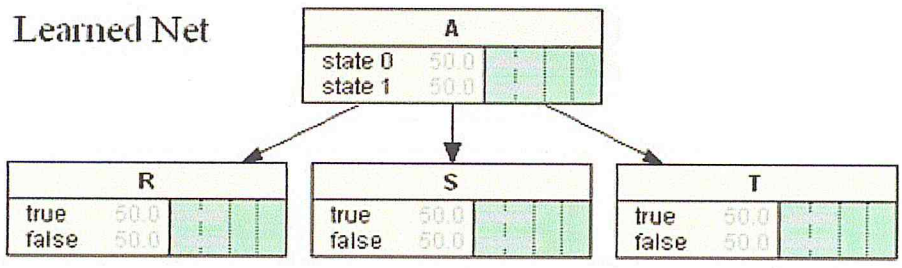

Figure 41 : exemple d'une structure d'un réseau bayésien

Prenons le nœud A, ses états sont state 0 et state 1, leurs valeurs sont 50 et 50.

La sortie de l'algorithme K2 sera un réseau bayésien optimisé.

La figure suivante montre les étapes de l'exécution de l'algorithme K2 :

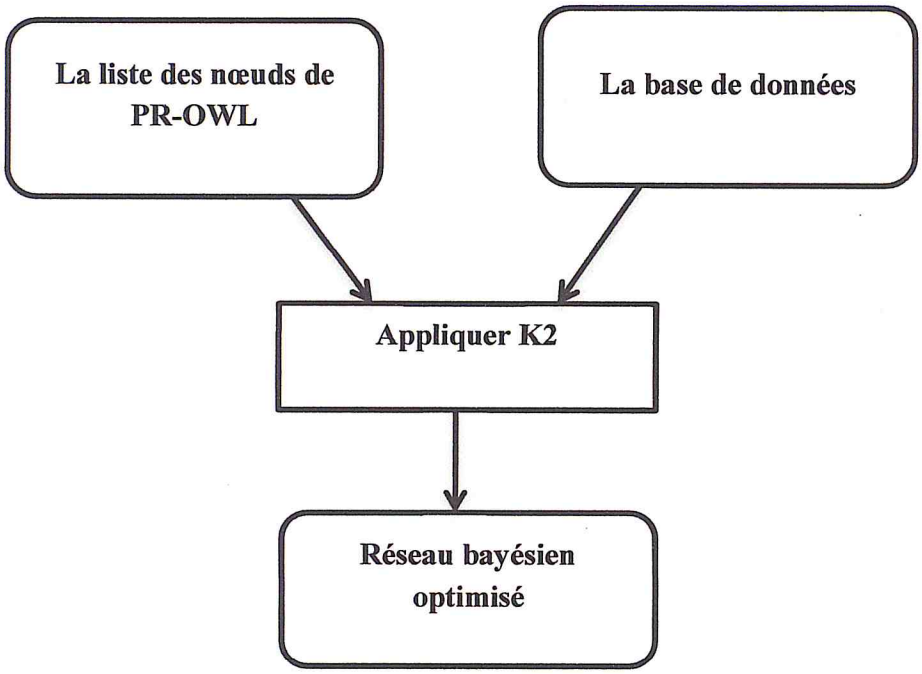

Figure 42 : étapes de l'exécution de l'algorithme K2

2.7 Application de l'algorithme EM :

l'apprentissage EM est une des techniques d'apprentissage des variables totalement "caché" (aussi connu comme «latentes»), qui sont, des variables pour lesquelles on n'a pas d'observations, mais on soupçonne qu'elle existe et peut être utile pour la modélisation du monde étudié.

EM travaille par un processus itératif, dans lequel on commence par un réseau de candidats, rapporte leurs log-vraisemblances, puis traite tous les paramètres du réseau pour trouver un meilleur réseau, la valeur de la log-vraisemblance de chaque itération sera meilleur que la valeur qui la précède. Ce processus est répété jusqu'à ce que les numéros de la log-vraisemblance du réseau ne s'améliore pas assez (selon une tolérance qu'on peut spécifier), ou le nombre souhaité d'itérations a été atteint (aussi une quantité qu'on peut spécifier).

Une fois qu'on a la structure du réseau bayésien faite par l'algorithme K2 et la table des statistiques, l'algorithme EM va s'occuper des calculs des paramètres manquants de notre ontologie probabiliste pour qu'elle soit prête pour les inférences.

La figure suivante montre les étapes de l'exécution de l'algorithme EM :

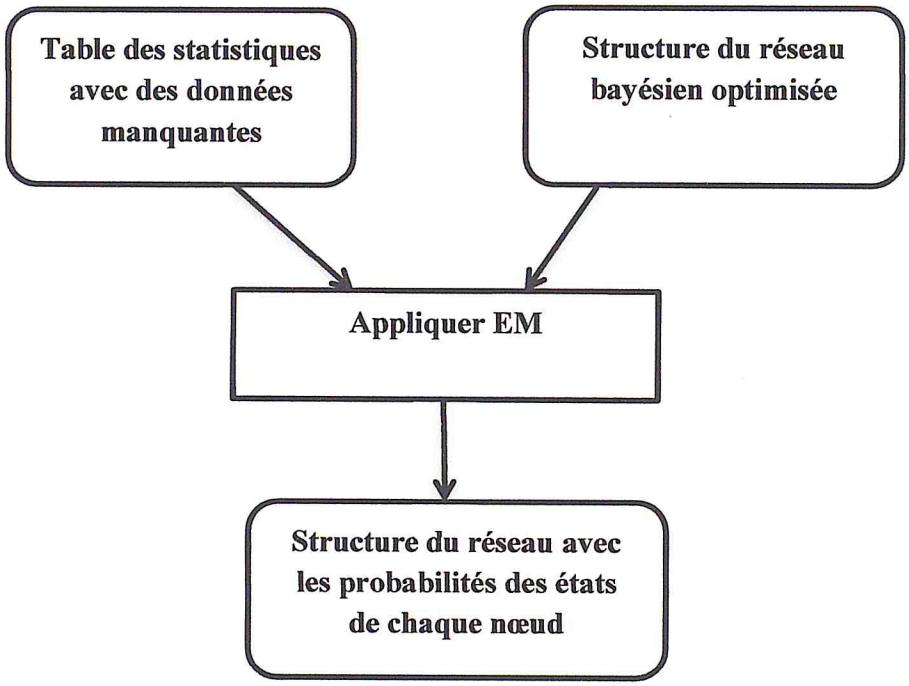

Figure 43 : étapes de l'exécution de l'algorithme EM

2.8 Une ontologie PR-OWL avec ses distributions :

Apres l'exécution de l'algorithme EM, il génère des valeurs de probabilité estimées pour les données manquantes à partir des statistiques faites selon les données dans l'ontologie formelle OWL.

Ces valeurs seront placées à la place de la déclaration des distributions dans les tables de l'ontologie probabiliste PR-OWL, donc ces valeurs sont générées automatiquement à partir de l'ontologie formelle, ce qui n'existait pas dans le langage PR-OWL.

Apres l'intégration des distributions de probabilités des nœuds, on pourra effectuer les inférences qu'on a besoin pour éclaircir les données incertaines du domaine étudié.

La figure ci-dessous montre une déclaration des distributions dans un fichier PR-OWL :

```
7948    </Literal>
7949        </DataPropertyAssertion>
7950        <DataPropertyAssertion>
7951            <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasDeclaration"/>
7952            <NamedIndividual IRI="http://www.pr-owl.org/pr-owl2.owl#isMemberOfCommittee_Table"/>
7953            <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">[
7954        true = .0001,
7955        false = .9999
7956    ]
7957    </Literal>
7958        </DataPropertyAssertion>
7959        <DataPropertyAssertion>
7960            <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasDeclaration"/>
7961            <NamedIndividual IRI="http://www.pr-owl.org/pr-owl2.owl#isParticipantIn_Table"/>
7962            <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">[
7963        true = .0001,
7964        false = .9999
7965    ]
7966    </Literal>
7967        </DataPropertyAssertion>
7968        <DataPropertyAssertion>
7969            <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasDeclaration"/>
7970            <NamedIndividual IRI="http://www.pr-owl.org/pr-owl2.owl#isProcurementFinished_Table"/>
7971            <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">[
7972        true = .5,
7973        false = .5
7974    ]
7975    </Literal>
```

Figure 44 : Exemple d'une déclaration d'une distribution d'une table PR-OWL

8. Conclusion :

Dans ce chapitre, Nous avons étudié les différentes étapes du processus de la construction d'une ontologie probabiliste PR-OWL d'un domaine donné, et aussi nous avons proposé une méthode pour automatiser l'apprentissage de la structure du réseau d'entré de l'algorithme EM qui sert à remplir les paramètres de la table PR-OWL d'une maniéré automatique, à la fin nous plaçons les valeurs obtenus à partir de ces algorithmes dans leurs places dans l'ontologie PR-OWL pour servir aux inférences ultérieurement.

Chapitre 5 : Implémentation et test du système

1. Introduction

Implémenter c'est effectuer l'ensemble des opérations qui permettent de définir un projet et de le réaliser, la conception et la mise en service du système ou du produit.

Dans ce chapitre nous allons décrire de façon visuelle l'implémentation de notre système, en effectuant des captures d'écran des différentes interfaces du système, et nous allons le tester en effectuant des captures sur les résultats de notre application.

2. Outils de développement de notre système

- **NetBeans et JDK**

Lors d'un développement d'application ou de la création d'un site web, il est plus qu'indispensable d'avoir un outil comme NetBeans IDE pour simplifier la tâche. NetBeans IDE « intégré » (en anglais, pour Integrated Development Environment, le produit d'un outil peut servir de matière première pour un autre outil **[52]**) n'est autre qu'un outil de développement professionnel. Il offre de nombreuses fonctionnalités pratiques. Ainsi qu'un environnement «Java Development Kit » JDK est requis pour les développements en Java de notre système.

- **JENA**

Jena est une API Java qui peut être utilisée pour créer et manipuler des graphes RDF et les ontologies exprimé en OWL. Jena possède des classes pour représenter des graphes, des ressources, des propriétés et des littéraux. Dans Jena, un graphe est appelé un modèle et est représenté par l'interface Model.

- **Netica API :**

  Développé depuis 1992 et commercialisé depuis 1995 par la société canadienne norsys (http://www.norsys.com) basé à Vancouver, le logiciel de réseaux bayésiens Netica est actuellement l'un des plus diffusés à l'échelle mondiale. Netica est utilisé pour le diagnostic, la précision ou la simulation dans les domaines de la finance, de l'environnement, de la médicine, de l'industrie et dans un grand nombre d'application nécessitant de raisonner en univers incertain. Une version gratuite de logiciel entièrement fonctionnelle est téléchargeable sur le site internet de Norsys.

  Netica permet l'apprentissage de table de probabilités à partir de donnée, au moyen d'un algorithme d'apprentissage bayésien. L'ensemble des tables de probabilités d'un

réseau bayésien peuvent donc être spécifiées en introduisant une base de données ou un échantillon de cas, de taille suffisamment grande.

Un algorithme d'apprentissage de structure sera prochainement disponible dans le logiciel **[34].**

L'API Netica-J est une bibliothèque complète des classes Java pour travailler avec les réseaux bayésiens (également connu sous le nom des réseaux de croyances, des modèles graphiques ou des modèles causaux probabilistes) et des diagrammes d'influence (Également connu sous le nom de réseaux de décision). Elle contient des fonctions pour construire, apprentissage des données, modifier, transformer, test de performance, enregistrer et lire les réseaux, ainsi qu'un moteur d'inférence puissant.

- **WEKA API :**

Weka est un ensemble d'algorithmes d'apprentissage automatique pour les tâches de data mining. Les algorithmes peuvent soit être appliqués directement à un ensemble de données ou appelées à partir de votre propre code Java. Weka contient des outils pour le prétraitement des données, la classification, la régression, le clustering, les règles d'ass ociation, et la visualisation. Il est également bien adapté pour le développement de nouveaux systèmes d'apprentissage automatique. Weka est un logiciel open source publié sous la licence GNU General Public License.

3. La mise en œuvre du système :

La première étape vers la construction d'une ontologie probabiliste, est d'importer un fichier OWL contenant les classes de PROWL, ses sous-classes et ses propriétés à notre projet à chaque fois qu'on veut créer une nouvelle ontologie probabiliste.

```
1   <?xml version="1.0"?>
2   <Ontology xmlns="http://www.w3.org/2002/07/owl#"
3        xml:base="file:/E:/MEBN/Ontologias/CGU/FraudInPublicProcurement.owl"
4        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
6        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
7        xmlns:xml="http://www.w3.org/XML/1998/namespace"
8        ontologyIRI="file:/E:/MEBN/Ontologias/CGU/FraudInPublicProcurement.owl">
9       <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
10      <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
11      <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
12      <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
13      <Import>http://www.pr-owl.org/pr-owl2.owl</Import>
```

Figure 45: Exemple d'importation du fichier PR-OWL

3.1 L'interface d'entrée :

La première interface à afficher est la suivante :

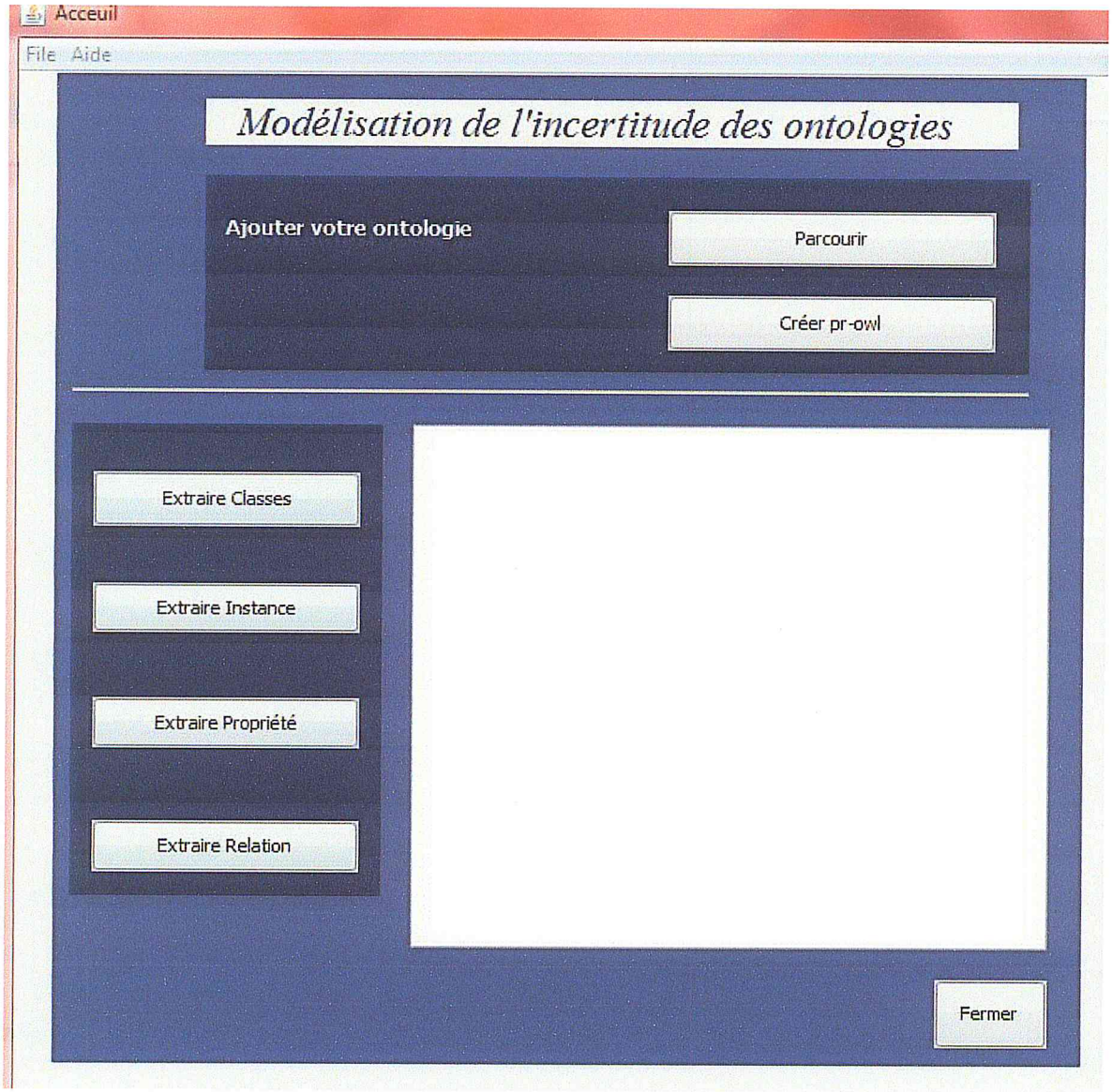

Figure 46 : l'interface d'entrées

C'est l'interface qui permet l'accès à la modélisation de l'ontologie. On doit d'abord importer l'ontologie OWL depuis un emplacement quelconque, et puis, il faut extraire les parties TBOX et ABOX de l'ontologie pour pouvoir accéder à l'interface de la création de l'ontologie probabiliste.

- **Parcourir :** elle importe le fichier de l'ontologie OWL en parcourant son chemin :

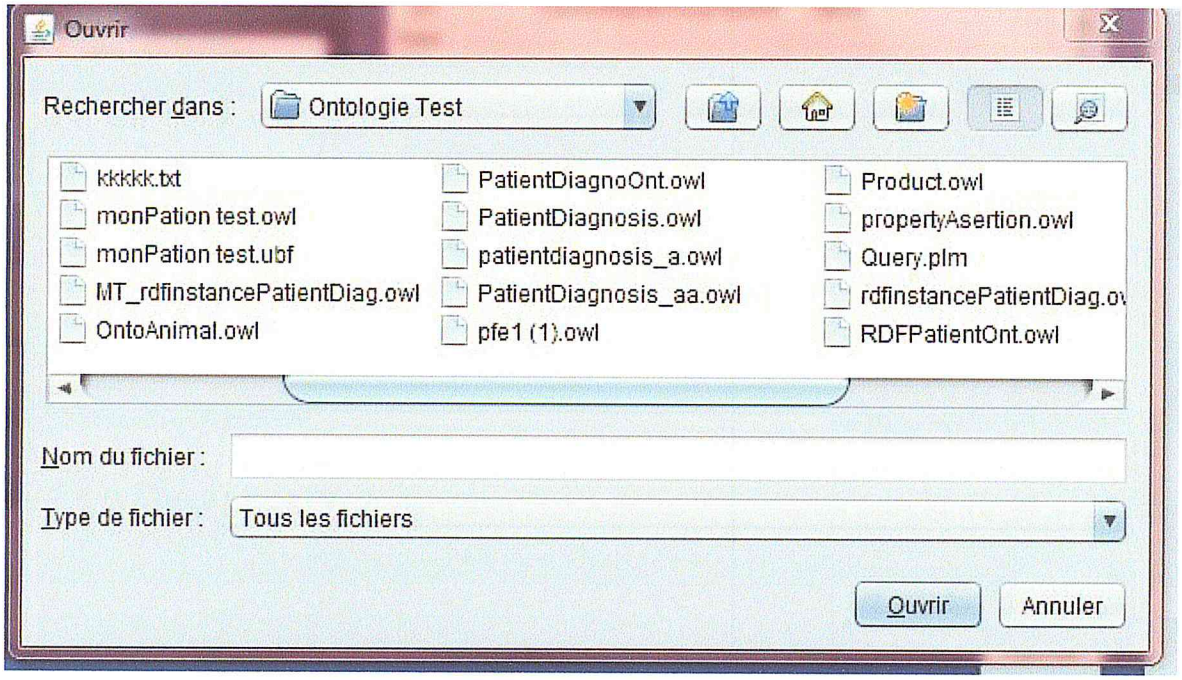

Figure 47 : choisir le fichier OWL de l'ontologie

- **Extraire classes :** elle fait l'extraction des classes de l'ontologie OWL, pour les utiliser ultérieurement dans la création des variables ordinaires.
- **Extraire instances :** elle fait l'extraction des instances de l'ontologie OWL, pour qu'on les utilise dans le choix des états des nœuds.
- **Extraire propriétés :** elle fait l'extraction des propriétés de l'ontologie (les deux types de propriétés : DataTypeProperty et ObjectProperty) pour la création de nœuds.
- **Extraire Relations :** elle fait l'extraction des instances avec leurs propriétés (DataTypeProperty ou ObjectProperty)
- **Créer PR-OWL :** Apres l'extraction des TBOX et ABOX, ça nous donne l'accès à l'interface suivante :

3.1 Interface de la gestion de PR-OWL :

Cette interface permet l'effectuation des différentes taches de notre système :

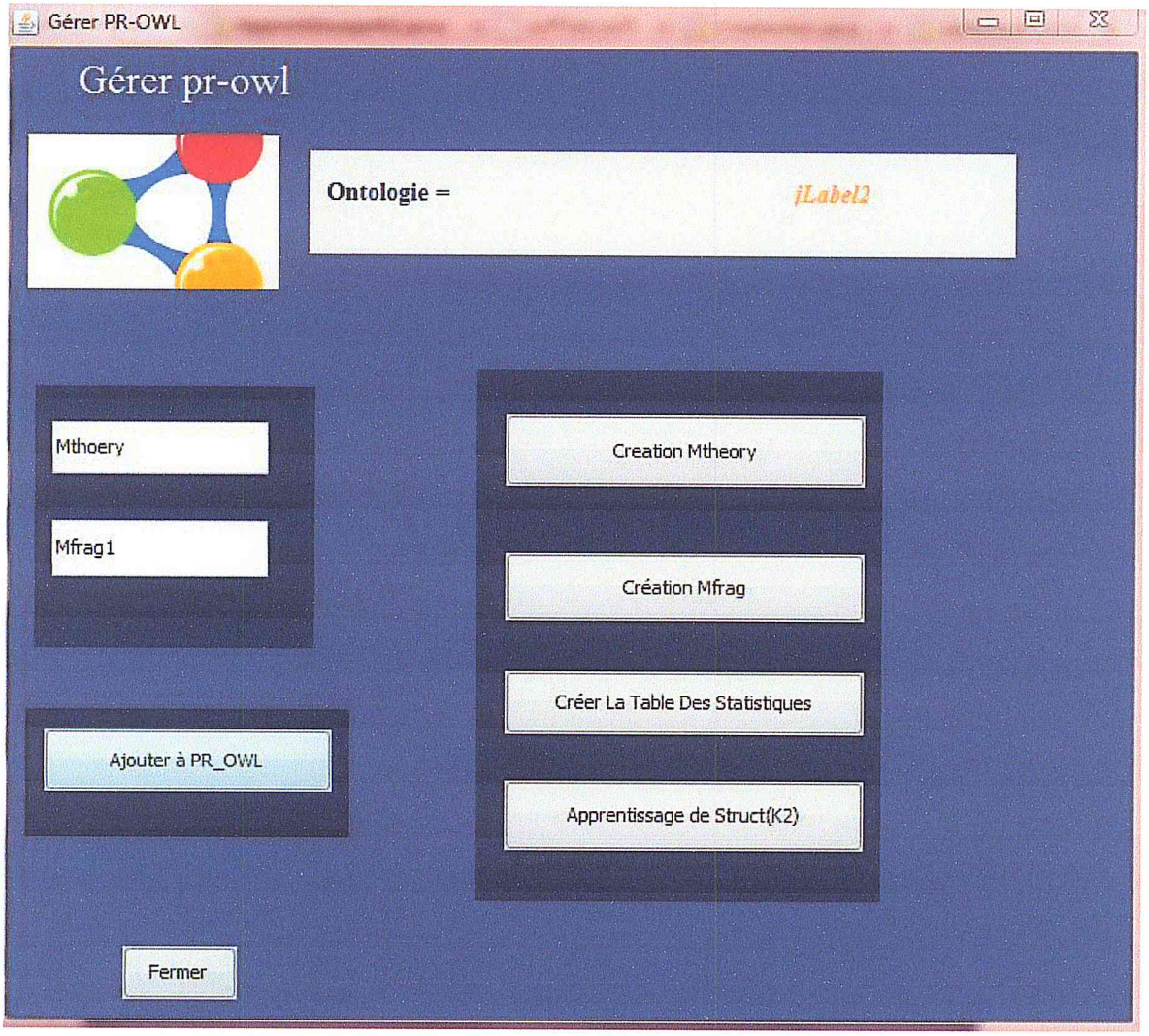

Figure 48 : interface de la gestion de PR-OWL

Elle permet de créer la MTeory , une fois qu'on entre le nom de la MTheory , on aura la main pour créer ses MFrags, et à la fin de la création de chaque MFrag on aura la possibilité de revenir à cette interface pour la création d'un autre MFrag ou la construction de la table des statistiques ou pour appliquer les algorithmes d'apprentissage.

3.1.1 Interface de la création d'un MFrag avec ses nœuds :

Cette interface permet la création des variables ordinaires ainsi que les trois types de nœuds (nœuds résidents, nœuds d'entrées, nœud d'input) dans un MFrag donnés dans l'interface précédente.

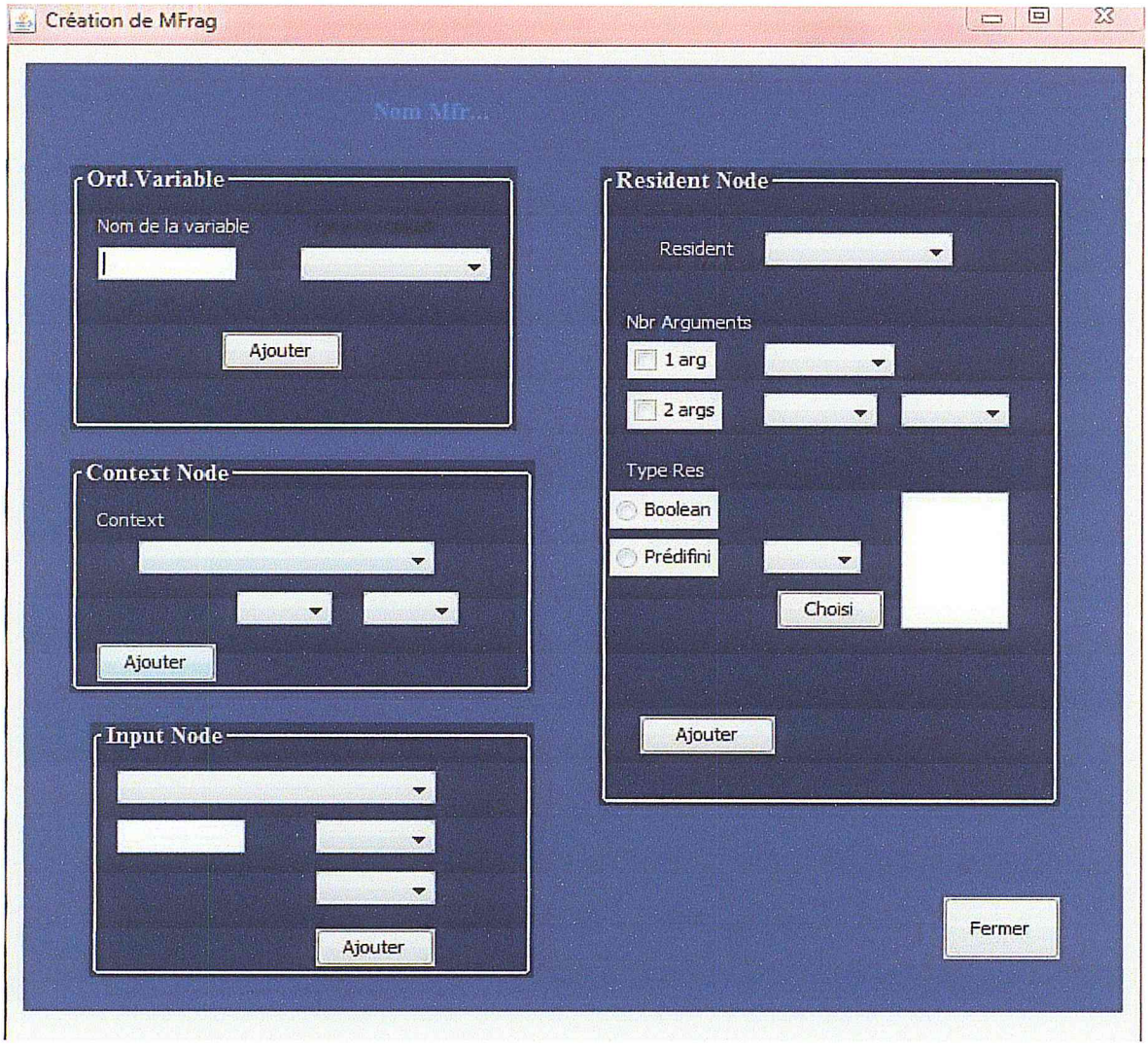

Figure 49 : Interface de la création d'un MFrag avec ses nœuds.

- Pour l'ajout de la variable ordinaire, on doit saisir le nom de la variable et choisir la classe dont lequel appartient cette variable.
- Pour l'ajout du nœud résident, on choisit à partir des propriétés de l'ontologie formelle la propriété qui représente une incertitude ou qui contribue à la modélisation de l'ontologie probabiliste. On doit sélectionner les arguments du nœud, et définir son état (booléen ou prédéfini).

- Pour l'ajout du nœud de contexte, on doit choisir parmi les nœuds crée déjà et définir ses arguments.
- Pour l'ajout des nœuds d'entrées, on doit spécifier le MFrag dont lequel il réside ce nœud d'entrées, après on choisit ses arguments.
- Une fois qu'on termine la création d'un MFrag on aura la main pour ajouter un autre MFrag ou de construire la table des statistiques pour l'algorithme EM,la base de données de l'algorithme K2 ou appliquer les algorithmes qui se trouve dans l'interface précédente.

3.1.2 Apprentissage de structure (Appliquer K2) :

Ce bouton nous prend à une autre interface pour créer la base de données et l'extraction des nœuds du PR-OWL :

3.1.2.2 Interface de la création de la base de données et l'application de l'algorithme K2 :

Cette interface nous permet de créer la base de données et appliquer l'algorithme K2 en utilisant cette base de données et la liste ordonnée des nœuds extraits auparavant.

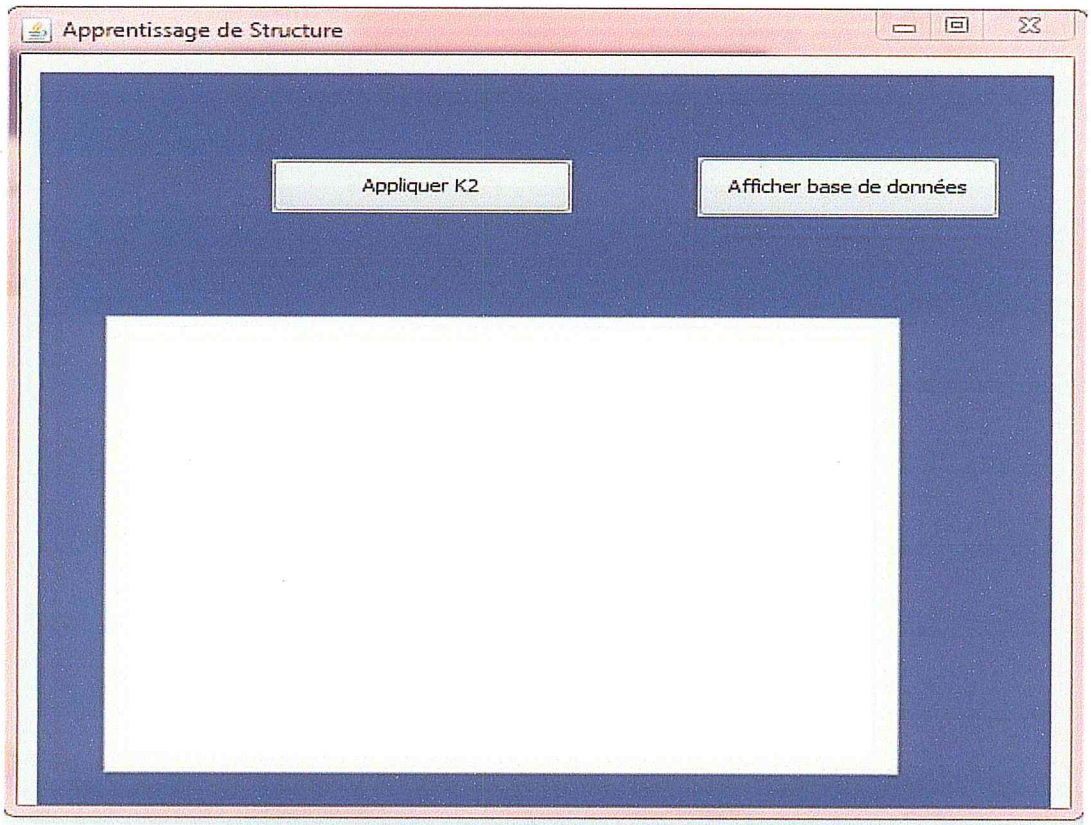

Figure 50 : interface de la création de la base de données

- **Afficher la base de données :** elle nous permet de créer et afficher la base de données, la base de donnée va être enregistrée sous forma .DNE, qui sera un fichier d'entrée de l'algorithme K2.
- **Appliquer K2 :** ça nous permet d'appliquer l'algorithme K2 sur la base de données créée, et la liste des nœuds de PR-OWL.

3.1.3 Apprentissage de paramètres :

Ce bouton nous prend à une interface pour la création d'une structure initiale du réseau :

3.1.3.1 Interface de l'extraction de la structure initiale du réseau :

Cette interface nous permet de construire une version non optimisée du réseau, mais cela n'est pas un souci car on en a besoin juste pour remplir la table des statistiques de l'algorithme EM.

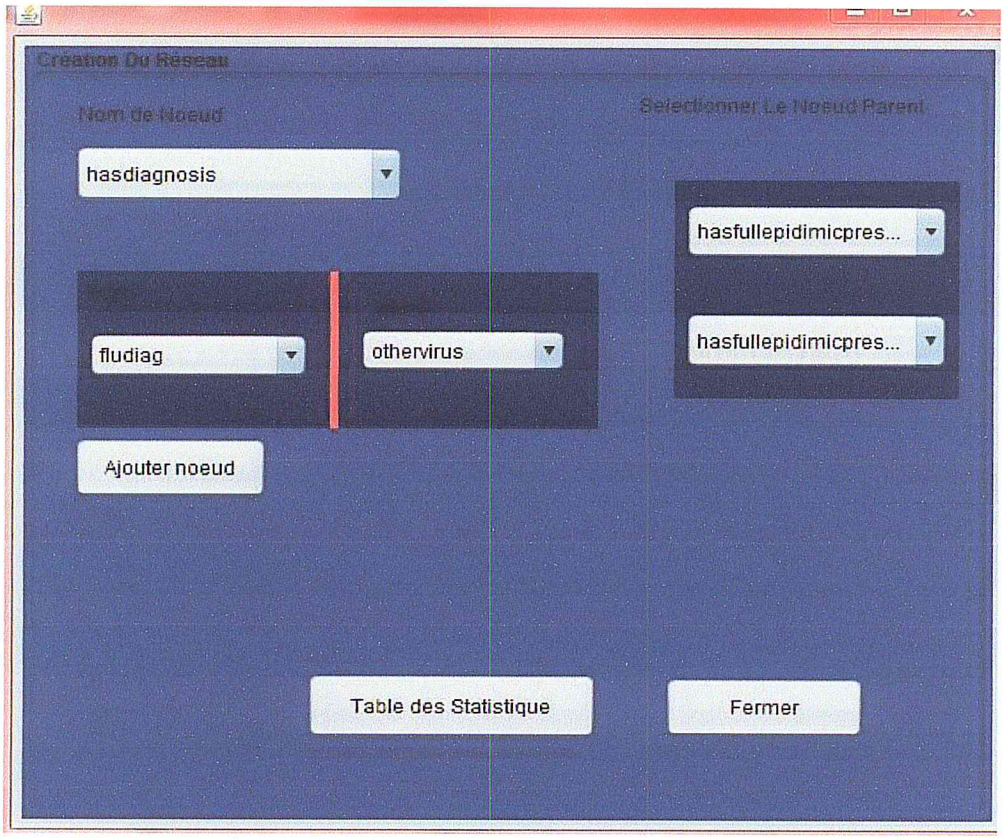

Figure 51 : Interface de l'extraction de la structure initiale du réseau

- **Ajouter nœud:** D'abord, on ajoute les nœuds en choisissant leurs états, et pour chaque nœud on choisit ses parents.
- **Table des Statistique:** Une fois le réseau initial est construit, on appuie sur ce bouton pour créer et afficher la table des statistiques.

3.1.3.1 Interface pour la création de la table des statistiques pour l'algorithme EM:

Cette interface permet de créer et d'afficher la table des statistiques qui contient des valeurs manquantes, elle va être enregistrée sous format d'un fichier CAS, qui est un fichier d'entrée de l'algorithme EM.

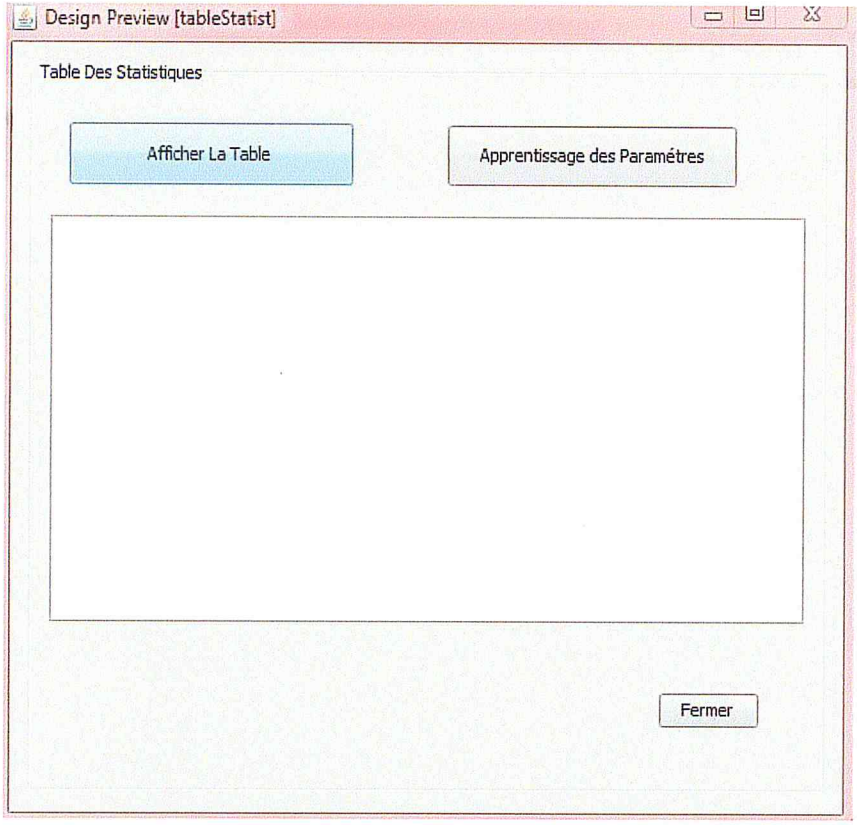

Figure 52 : interface de l'affichage de la table des statistiques pour l'algorithme EM

- **Afficher Table :** ça permet de créer et d'afficher la table des statistiques.
- **Apprentissage de parametres :** ça permet d'appliquer l'algorithme EM, il a comme entrée le fichier CAS de la table des statistiques et le fichier DNE de la structure optimisée par l'algorithme K2.

3.1.4 Ajouter à PR-OWL :

Lorsqu'on termine avec l'apprentissage des paramètres, les valeurs estimées des données manquantes vont être placées automatiquement dans la table du PR-OWL, pour ce faire, nous importons les valeurs de probabilités estimées à une assertion d'une propriété de donnée de l'ontologie PR-OWL.

4. Test:

Dans cette dernière partie nous allons effectuer un test sur notre système implémenté.

La méthode que nous avons utilisée pour tester notre système c'est d'effectuer les étapes de notre modélisation pour créer une ontologie probabiliste utilisable pour soutenir le diagnostic d'un patient entrant dans une clinique avec des symptômes. S'il a récemment visité une zone avec une épidémie de grippe, et qui pourrait être la cause. Sinon, le patient présente un autre type de virus.

4.1 Résultat de la création de l'ontologie probabiliste :

Apres l'effectuation des étapes de la création de l'ontologie probabiliste on a eu un fichier OWL qui porte la MTheory avec ses différents MFrags du domaine.

4.1.1 Comparaison avec l'ontologie formelle et l'ontologie probabiliste obtenue:

La hiérarchie des classes avant PR-OWL :

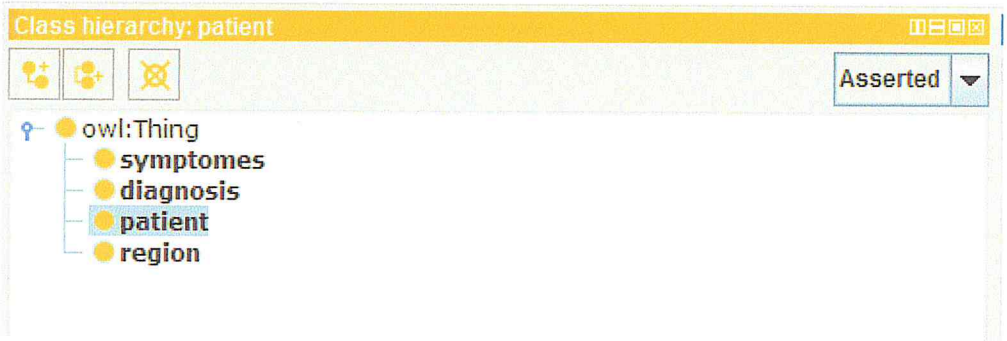

Figure 53 : hiérarchie des classes avant PR-OWL

La hiérarchie de classes après PR-OWL :

Figure 54 : hiérarchie des classes après PR-OWL

La hiérarchie des propriétés d'objet avant PR-OWL :

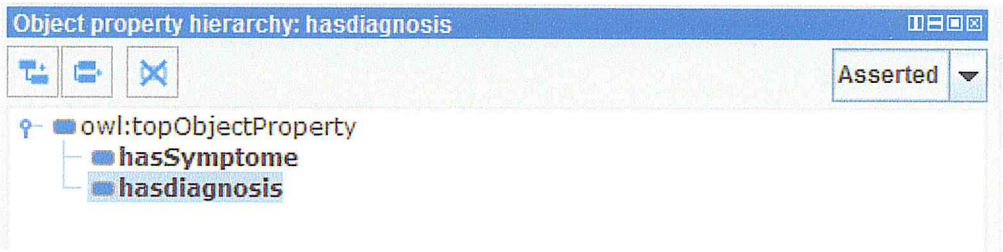

Figure 55 : La hiérarchie des propriétés d'objet avant PR-OWL

La hiérarchie des propriétés d'objet après PR-OWL :

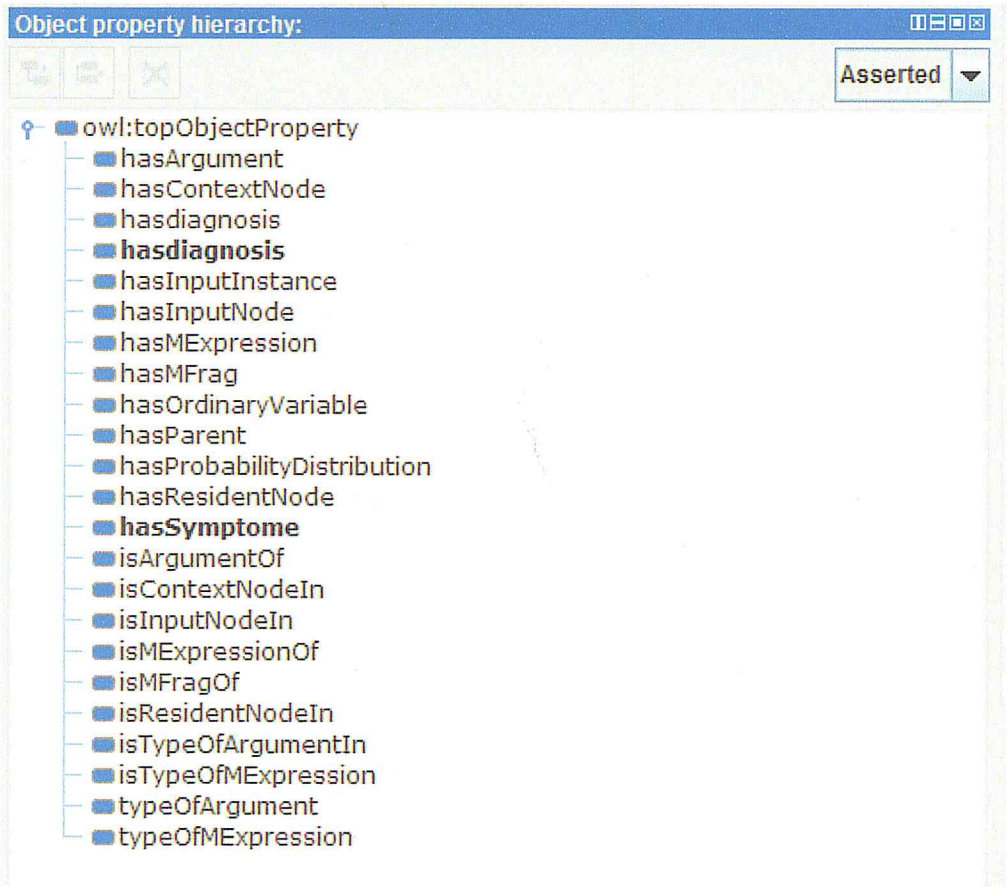

Figure 56: La hiérarchie des propriétés d'objet après PR-OWL

La hiérarchie des propriétés de données avant PR-OWL :

Figure 57 : La hiérarchie des propriétés de données avant PR-OWL

La hiérarchie des propriétés de données après PR-OWL :

Figure 58 : La hiérarchie des propriétés de données avant PR-OWL

4.1.2 L'ontologie probabiliste obtenue :

L'ontologie probabiliste obtenue est la combinaison de l'ontologie formelle avec la syntaxe de PR-OWL.

- **MTeory :**
  La syntaxe PR-OWL de l'ajout d'une MTheory est la suivante :

```
<ClassAssertion>
   <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MTheory"/>
   <NamedIndividual IRI="#patientdiagnosis"/>
</ClassAssertion>
```

Figure 59 : la MTeory

- **MFrag :**

  La syntaxe PR-OWL de l'ajout d'un MFrag est la suivante :

```
<ObjectPropertyAssertion>
   <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMFragOf"/>
   <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
   <NamedIndividual IRI="#patientdiagnosis"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
   <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasMFrag"/>
    <NamedIndividual IRI="#patientdiagnosis"/>
    <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
</ObjectPropertyAssertion>
```

Figure 60 : Le MFrag

➢ **Variables ordinaires :**

La syntaxe PR-OWL de l'ajout d'une variable ordinaire est la suivante :

```
<AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
    <IRI>#MFrag1.rgn</IRI>
    <Literal datatypeIRI="&xsd;string">OX1</Literal>
</AnnotationAssertion>
<ObjectPropertyAssertion>
   <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasOrdinaryVariable"/>
   <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
    <NamedIndividual IRI="#MFrag1.rgn"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
   <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isSubstitutedBy"/>
   <NamedIndividual IRI="#MFrag1.rgn"/>
   <Literal datatypeIRI="&xsd;anyURI">http://www.semanticweb.org/patientdiagnosis.owl#region</Literal>
</DataPropertyAssertion>
```

Figure 61: variables ordinaires

➢ **Nœud résident :**

La syntaxe PR-OWL de l'ajout d'un nœud résident est la suivante :

```
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#DeclarativeDistribution"/>
    <NamedIndividual IRI="http://www.pr-owl.org/pr-owl2.owl#hasfullepidimicpresent_Table"/>
</ClassAssertion>

<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#DomainResidentNode"/>
    <NamedIndividual IRI="#Domain_Res.hasfullepidimicpresent"/>
</ClassAssertion>

<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#SimpleMExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_hasfullepidimicpresent"/>
</ClassAssertion>

<ClassAssertion>
   <Class IRI="http://www.pr-owl.org/pr-owl2.owl#RandomVariable"/>
    <NamedIndividual IRI="#RV_hasfullepidimicpresent"/>
</ClassAssertion>

<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MappingArgument"/>
    <NamedIndividual IRI="#RV_hasfullepidimicpresent_1"/>
</ClassAssertion>

<ClassAssertion>
   <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariableArgument"/>
    <NamedIndividual IRI="#hasfullepidimicpresent_1"/>
</ClassAssertion>
```

```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasMExpression"/>
    <NamedIndividual IRI="#Domain_Res.hasfullepidimicpresent"/>
    <NamedIndividual IRI="#MEXPRESSION_hasfullepidimicpresent"/>
</ObjectPropertyAssertion>

<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isResidentNodeIn"/>
    <NamedIndividual IRI="#Domain_Res.hasfullepidimicpresent"/>
    <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
</ObjectPropertyAssertion>

<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_hasfullepidimicpresent"/>
    <NamedIndividual IRI="#hasfullepidimicpresent_1"/>
</ObjectPropertyAssertion>

<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMExpressionOf"/>
    <NamedIndividual IRI="#MEXPRESSION_hasfullepidimicpresent"/>
    <NamedIndividual IRI="#Domain_Res.hasfullepidimicpresent"/>
</ObjectPropertyAssertion>

<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfMExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_hasfullepidimicpresent"/>
    <NamedIndividual IRI="#RV_hasfullepidimicpresent"/>
</ObjectPropertyAssertion>

<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasResidentNode"/>
    <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
    <NamedIndividual IRI="#Domain_Res.hasfullepidimicpresent"/>
</ObjectPropertyAssertion>

<AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
    <IRI>#Domain_Res.hasfullepidimicpresent</IRI>
    <Literal datatypeIRI="&xsd;string">hasfullepidimicpresent</Literal>
</AnnotationAssertion>

<DataPropertyAssertion>
    <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgumentNumber"/>
    <NamedIndividual IRI="#hasfullepidimicpresent_1"/>
    <Literal datatypeIRI="&xsd;integer">1</Literal>
</DataPropertyAssertion>

<DataPropertyAssertion>
    <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgumentNumber"/>
    <NamedIndividual IRI="#RV_hasfullepidimicpresent_1"/>
    <Literal datatypeIRI="&xsd;integer">1</Literal>
</DataPropertyAssertion>

<DataPropertyAssertion>
    <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#definesUncertaintyOf"/>
    <NamedIndividual IRI="#RV_hasfullepidimicpresent"/>
    <Literal datatypeIRI="&xsd;anyURI">http://www.semanticweb.org/patientdiagnosis.owl#hasfullepidimicpresent</Literal>
</DataPropertyAssertion>
```

```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfArgument"/>
    <NamedIndividual IRI="#hasfullepidimicpresent_1"/>
    <NamedIndividual IRI="#MFrag1.rgn"/>
</ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
     <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
      <NamedIndividual IRI="#RV_hasfullepidimicpresent"/>
      <NamedIndividual IRI="#RV_hasfullepidimicpresent_1"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
      <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfMExpression"/>
      <NamedIndividual IRI="#RV_hasfullepidimicpresent"/>
      <NamedIndividual IRI="#MEXPRESSION_hasfullepidimicpresent"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
     <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
      <NamedIndividual IRI="#RV_hasfullepidimicpresent_1"/>
      <NamedIndividual IRI="#RV_hasfullepidimicpresent"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
     <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfArgumentIn"/>
      <NamedIndividual IRI="#MFrag1.rgn"/>
      <NamedIndividual IRI="#hasfullepidimicpresent_1"/>
  </ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
      <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
      <NamedIndividual IRI="#hasfullepidimicpresent_1"/>
      <NamedIndividual IRI="#MEXPRESSION_hasfullepidimicpresent"/>
  </ObjectPropertyAssertion>
```

Figure 62 : Un nœud résident

- **Nœud d'entrée :**

  La syntaxe PR-OWL de l'ajout d'un nœud d'entrée est la suivante :

```
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#GenerativeInputNode"/>
    <NamedIndividual IRI="#IX1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariableArgument"/
    <NamedIndividual IRI="#IX1_1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_IX1"/>
</ClassAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasInputNode"/>
    <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
    <NamedIndividual IRI="#IX1"/>
</ObjectPropertyAssertion>

  <ObjectPropertyAssertion>
     <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasInputInstance"/>
      <NamedIndividual IRI="#Domain_Res.hasfullepidimicpresent"/>
      <NamedIndividual IRI="#IX1"/>
  </ObjectPropertyAssertion>
```

```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isInputNodeIn"/>
    <NamedIndividual IRI="#IX1"/>
    <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
    <NamedIndividual IRI="#IX1_1"/>
    <NamedIndividual IRI="#MEXPRESSION_IX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfArgument"/>
    <NamedIndividual IRI="#IX1_1"/>
    <NamedIndividual IRI="#region_mf.rgn"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_IX1"/>
    <NamedIndividual IRI="#IX1_1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMExpressionOf"/>
    <NamedIndividual IRI="#MEXPRESSION_IX1"/>
    <NamedIndividual IRI="#IX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfMExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_IX1"/>
    <NamedIndividual IRI="#RV_hasfullepidimicpresent"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfMExpression"/>
    <NamedIndividual IRI="#RV_hasfullepidimicpresent"/>
    <NamedIndividual IRI="#MEXPRESSION_IX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfArgumentIn"/>
    <NamedIndividual IRI="#region_mf.rgn"/>
    <NamedIndividual IRI="#IX1_1"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgumentNumber"/>
    <NamedIndividual IRI="#IX1_1"/>
    <Literal datatypeIRI="&xsd;integer">1</Literal>
</DataPropertyAssertion>
<AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
    <IRI>#IX1</IRI>
    <Literal datatypeIRI="&xsd;string">IX1</Literal>
</AnnotationAssertion>
```

Figure 63 : un nœud d'entrée

➢ **Nœud de contexte :**

La syntaxe PR-OWL de l'ajout d'un nœud de contexte est la suivante :

```
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#ContextNode"/>
    <NamedIndividual IRI="#CX1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#BooleanMExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#MExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariableArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_1"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="http://www.pr-owl.org/pr-owl2.owl#OrdinaryVariableArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_2"/>
</ClassAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasMExpression"/>
    <NamedIndividual IRI="#CX1"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isContextNodeIn"/>
    <NamedIndividual IRI="#CX1"/>
    <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasContextNode"/>
    <NamedIndividual IRI="#Domain_MFrag.MFrag1"/>
    <NamedIndividual IRI="#CX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_2"/>
</ObjectPropertyAssertion>
```

```
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isMExpressionOf"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
    <NamedIndividual IRI="#CX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfMExpression"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
    <NamedIndividual IRI="#RV_hasvisitedregion"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_1"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_1"/>
    <NamedIndividual IRI="#MFrag1.p"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isArgumentOf"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_2"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#typeOfArgument"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_2"/>
    <NamedIndividual IRI="#region_mf.rgn"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfMExpression"/>
    <NamedIndividual IRI="#RV_hasvisitedregion"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfArgumentIn"/>
    <NamedIndividual IRI="#MFrag1.p"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_1"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="http://www.pr-owl.org/pr-owl2.owl#isTypeOfArgumentIn"/>
    <NamedIndividual IRI="#region_mf.rgn"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_2"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgumentNumber"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_1"/>
    <Literal datatypeIRI="&xsd;integer">1</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasArgumentNumber"/>
    <NamedIndividual IRI="#MEXPRESSION_CX1_2"/>
    <Literal datatypeIRI="&xsd;integer">2</Literal>
</DataPropertyAssertion>
<AnnotationAssertion>
    <AnnotationProperty abbreviatedIRI="rdfs:comment"/>
    <IRI>#CX1</IRI>
    <Literal datatypeIRI="&xsd;string">CX1</Literal>
</AnnotationAssertion>
```

Figure 64 : Un nœud de contexte

4.2 Résultat de la création de la base de données:

Apres la création de la base de données, nous avons obtenu le fichier ARFF suivant :

```
@relation test

@attribute hasfullepidimicpresent {0,1}
@attribute hasSymptom   {0,1,2}
@attribute hasAge   {0,1,2,3}
@attribute hasdiagnosis  {0,1 }

@data
0,1,0,0
1,0,1,1
1,1,1,0
1,2,1,0
0,0,0,1
0,1,0,0
1,0,1,1
1,1,1,0
1,2,1,0
0,0,0,1
0,1,0,0
1,0,1,1
1,1,1,0
1,2,1,0
0,0,0,1
0,1,0,0
1,0,1,1
1,1,1,0
1,2,1,0
0,0,0,1
0,1,0,0
1,0,1,1
1,1,1,0
1,2,1,0
0,0,0,1
0,1,0,0
1,0,1,1
1,1,1,0
1,2,1,0
0,0,0,1
```

Figure 65 : Résultat de la création de la base de données

4.2.1 Résultat de l'algorithme K2 :

Le résultat de l'algorithme K2 est le fichier DNE suivant :

```
bnet testo {
AutoCompile = TRUE;
autoupdate = TRUE;
whenchanged = 1236956075;

visual V1 {
    defdispform = BELIEFBARS;
    nodelabeling = TITLE;
    NodeMaxNumEntries = 50;
    nodefont = font {shape= "Arial"; size= 10;};
    linkfont = font {shape= "Arial"; size= 9;};
    windowposn = (44, 11, 1097, 733);
    resolution = 72;
    drawingbounds = (1080, 720);
    showpagebreaks = FALSE;
    usegrid = TRUE;
    gridspace = (6, 6);
    NodeSet Node {BuiltIn = 1; Color = 0xc0c0c0;};
    NodeSet Nature {BuiltIn = 1; Color = 0xf8eed2;};
    NodeSet Deterministic {BuiltIn = 1; Color = 0xd3caa6;};
    NodeSet Finding {BuiltIn = 1; Color = 0xc8c8c8;};
    NodeSet Constant {BuiltIn = 1; Color = 0xffffff;};
    NodeSet ConstantValue {BuiltIn = 1; Color = 0xfffffb4;};
    NodeSet Utility {BuiltIn = 1; Color = 0xffbdbd;};
    NodeSet Decision {BuiltIn = 1; Color = 0xdee8ff;};
    NodeSet Documentation {BuiltIn = 1; Color = 0xf0fafa;};
    NodeSet Title {BuiltIn = 1; Color = 0xffffff;};
    PrinterSetting A {
        margins = (1270, 1270, 1270, 1270);
        landscape = FALSE;
        magnify = 1;
        };
    };
node hasdiagnosis {
    kind = NATURE;
    discrete = TRUE;
    states = (fludiag, othervirus);
    parents = (hasfullepidimicpresent,hasSymptom,hasAge);
    };
  node hasfullepidimicpresent {
      kind = NATURE;
      discrete = TRUE;
      states = (false, true);
      parents = ();
      };
```

```
node hasSymptom {
    kind = NATURE;
    discrete = TRUE;
    states = (feiver, tatigue, othersymptom);
    parents = ();
    };
node hasAge {
    kind = NATURE;
    discrete = TRUE;
    states = (two,twelve,fifty,seventy);
    parents = ();
    };
};
```

Figure 66 : résultat de l'application de l'algorithme K2

La figure suivante représente cette structure :

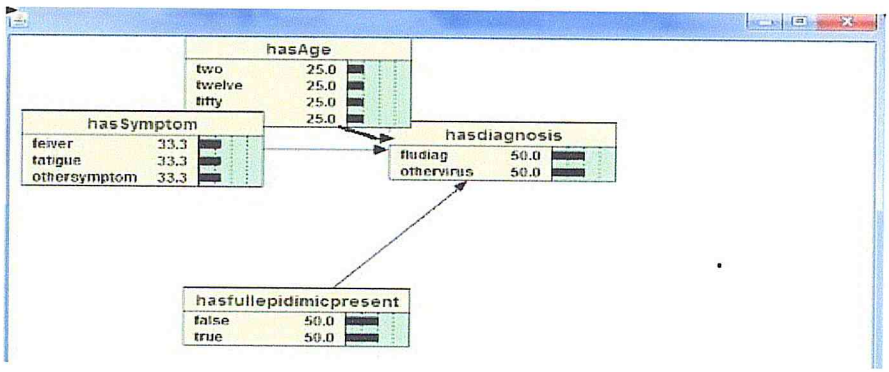

Figure 67 : la structure du réseau obtenu par l'algorithme K2

4.3 Résultat de la création de la table des statistiques :

Apres la création de la structure initiale du réseau on a créé la table des statistiques et on a obtenu le fichier CAS suivant :

```
//~->[DNET-1]->~
numCas  hasfullepidimicpresent  hasSymptome     hasdiagnosis        hasAge

0                    *              *               *               twenty
1                    *              *               *               two
2                   true            feiver          fludiag         fifty
3                   false           *               *               two
4                   true            *               *               *
5                   true           othersymptom      *              *
6                    *             fatigue          othervirus      twenty
7                    *              *               *               *
8                    *              *               *               *
9                    *             otherSymtom      fludiag         two
10                   *              *               *               *
11                   *              *               *               *
```

Figure 68 : Table des statistiques des données manquantes

4.3.1 Résultat de l'algorithme EM :

Une fois les deux entrées principales de l'algorithme EM ont été créées qui sont la structure initiale du réseau et la table des statistiques, nous avons appliqué l'algorithme EM et nous avons obtenu le fichier DNE suivant :

```
bnet New {
autoupdate = FALSE;

node hasfullepidimicpresent {
    kind = NATURE;
    discrete = TRUE;
    chance = CHANCE;
    states = (true, false);
    statetitles = ("hasfullepidimicpresent", );
    parents = ();
    probs =
        // true          false
          (0.5000284,   0.4999716);
    numcases = 4.00002;
    };

node hasSymptome {
    kind = NATURE;
    discrete = TRUE;
    chance = CHANCE;
    states = (feiver, fatigue);
    statetitles = ("hasSymptome", );
    parents = ();
    probs =
        // feiver         fatigue
          (0.5127234,   0.4872766);
    numcases = 4.00002;
    };

node hasdiagnosis {
    kind = NATURE;
    discrete = TRUE;
    chance = CHANCE;
    states = (fludiag, othervirus);
    statetitles = ("hasdiagnosis", );
    parents = (hasfullepidimicpresent, hasSymptome);
    probs =
        // fludiag        othervirus
          (0.04661103,  0.953389,
           0.999979,    2.107023e-5,
           0.05028192,  0.9497181,
           0.9999789,   2.10729e-5);
    numcases =

          (0.5255503,
           0.4746032,
           0.5253835,
           0.4745431);
    };
```

```
node hasAge {
    kind = NATURE;
    discrete = TRUE;
    chance = CHANCE;
    states = (two, twelve);
    statetitles = ("hasAge", );
    parents = ();
    probs =
        // two       twelve
          (0.0000045,          0.9999955);
    numcases = 2.00002;
    };
ElimOrder = (hasAge, hasfullepidimicpresent, hasSymptome, hasdiagnosis);
};
```

Figure 69 : Résultat de l'application de l'algorithme EM

4.4 Résultat de l'intégration des résultats de l'algorithme EM dans l'ontologie probabiliste PR-OWL :

Nous avons extrait les valeurs de l'algorithme EM et nous les avons placé dans la propriété de données (DataTypeProperty) *hasDeclaration* des table de probabilités des nœuds. Nous avons obtenu le résultat suivant :

```
    <DataProperty IRI="http://www.pr-owl.org/pr-owl2.owl#hasDeclaration"/>
    <NamedIndividual IRI="http://www.pr-owl.org/pr-owl2.owl#hasdiagnosis_Table"/>
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#string">if any diag have ( hasfullepidimicpresent = true ) [
  if any diag have ( hasSymptome= feiver ) [
    fludiag = 0.04661103,
    othervirus = 0.953389
  ] else [
    fludiag =  0.999979,
    othervirus = 2.107023e-5
  ]
] else if any diag have ( hasfullepidimicpresent  = false ) [
  if any diag have (  hasSymptome= feiver ) [
    fludiag =  0.05028192,
    othervirus = 0.9497181
  ] else [
    fludiag = 0.9999789,
    othervirus = 2.10729e-5
  ]
]
</Literal>
  </DataPropertyAssertion>
```

Figure 70 : Intégration des résultats de l'algorithme EM dans l'ontologie PR-OWL

5. Conclusion :

Dans ce chapitre nous avons montré l'implémentation de notre système de la modélisation de l'incertitude dans les ontologies avec des captures de notre application, et nous avons testé le système avec une petite ontologie de diagnostic.

Conclusion générale

Conclusion générale :

Les ontologies sont une approche puissante pour permettre le partage des connaissances et de soutenir l'interopérabilité entre les personnes et les systèmes informatiques. Cependant, les ontologies traditionnelles (formelles) ne fournissent pas un soutien adéquat à l'incertitude, une caractéristique fondamentale d'un environnement complexe dans un monde réel ouvert.

Pour faire face à tels problème important. Ce mémoire présente une méthode qui porte les procédures générales pour la modélisation de l'incertitude dans ontologies spécifiques à des domaines d'une façon automatique en utilisant MEBN/PR-OWL.

L'ontologie probabiliste, PR-OWL, en plus d'être compatible avec OWL, cette ontologie basée sur les réseaux bayésiens multi entités est capable de prédire tout en tenant compte des incertitudes autour des données.

Il est prouvé que les réseaux bayésiens sont des méthodes fiables pour modéliser l'incertain et pour faire des prédictions en se basant sur des faits réels même en présence d'incertitudes autour de ces faits. Nous avons conclu que les réseaux bayésiens peuvent être appliqués dans le web sémantique afin de résoudre le problème de l'incertitude.

Nous avons proposé pour cette modélisation un processus qui porte des étapes :

- **L'importation de l'ontologie formelle OWL :** sélectionner quelle ontologie qu'on veut traiter
- **Extraire les partie TBox et ABox :** extraire les classes, propriétés, les propriétés avec leurs domaines et rangs, et les assertions.
- **La création de la MTheory et les MFrags :** créer la MTheory et ses MFrags avec leurs nœuds selon le langage PR-OWL.
- **Application de l'algorithme K2 :** nous avons appliqué l'algorithme K2 afin d'optimiser la structure de notre réseau bayésien, cette structure va etre un parametre d'entrée de l'algorithme EM.
- **Application de l'algorithme EM sur les nœuds des MFrags :** une approche pour l'apprentissage des données manquantes (les nœuds) avec un algorithme (EM) basé sur les réseaux bayésiens qui arrive à faire une prédiction efficace sur les données manquantes.

  Ces prédictions sont sous forme des valeurs de probabilités qui vont être intégrées dans le ficher PR-OWL dans la section de la déclaration de la table PR-OWL afin de faire des inférences par la suite.

Nous avons implémenté notre système qui est générique et nous l'avons testé avec une ontologie d'un domaine médicale, ainsi nous avons montré les différentes interfaces nécessaires pour la construction de l'ontologie PR-OWL, et les interfaces nécessaires pour l'application des algorithmes que nous avons utilisé pour l'apprentissage de la structure K2 et l'apprentissage des paramètres manquant EM

References

References:

[1] Matsumoto, S., Carvalho, R., Costa, P., Laskey, K.B., Santos, L.L. and Ladeira, M. There's No More Need to be a Night OWL: on the PR-OWL for a MEBN Tool Before Nightfall. in Introduction to the Semantic Web: Concepts, Technologies and Applications, G. Fung, Ed. iConcept Press, 2011.

[2] BERNERS-LEE T. Standards, Semantics and Survival. SIIA Upgrade 2003: pp. 6–10.

[3] BOURHIS A, PERRIN N , 2010 .Etat de l'art sur les technologies du web sémantique, M2 MIAGE.

[4] Kenneth J. Laskey, et al,. 20/02/2016. W3C Uncertainty Reasoning for the World Wide Web Incubator Group [en ligne], Adressee URL: https://www.w3.org/2005/Incubator /urw3 /group/draftReport.html

[5] VALERY PSYCnHE, Proposition d'une méthode d'ingénierie ontologique pour les EIAH : application aux systèmes auteurs, Programme de doctorat en informatique, Université du Québec à Montréal Canada- (Mai 2004).

[6] Neeches R., Fikes R. E., Finin T., Gruber T. R., Senator T. et Swartout W. R., 1991. Enabling technology for knowledge sharing. AI Magazine. 12: 36- 56.

[7] Gruber T., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition Journal. 5:199 -220.

[8] Borst W. N., 1997. Construction of Engineering Ontologies. Thèse de doctorat, Center for Telematica and Information Technology, Université de Tweenty, Enschede.

[9] Studer R., Benjamins V. R. et Fensel D., 1998. Knowledge engineering: Principles and Methods. Data Knowledge Engineering, 25: 161-197.

[10] Guarino N., et Giaretta P., 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, Mars N. J. I., Amsterdam: IOS Press. pp. 25–32.

[11] Swartout B., Patil R., Knight K. et Russ T., 1997. Towards Distributed Use of Large Scale Ontologies. Spring Symposium Series on Ontological Engineering, Stanford University, CA. pp. 138-138.

[12] Gomez P.A., 1999. Développements récents en matière de conception, de maintenance et d'utilisation d'ontologies. 3èmes rencontres Terminologie et intelligence artificielle TIA. 19 :9-20.

[13] Gomez Pérez et Benjamins V R. (1999). Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. Proceedings of the IJCAI-99 workshop on Ontology and ProblemSolving Methods (KRR5).

[14] Staab S. et Maedche A., 2000. Axioms are objects too: Ontology engineering beyond the modeling of concepts and relations. Research report 399, Institute AIFB, Karlsruhe.

[15] Psyché V. (2007). Rôle des ontologies en ingénierie des EIAH : Cas d'un système d'assistance au design pédagogique. Thèse de doctorat, Université du Québec-Montréal.

[16] Mizoguchi R. et Ikeda M., 1996. Towards Ontological Engineering. Technical Report AI-TR-96-1, I.S.I.R., Osaka University, Japan.

[17] Bachimont B. (2000). Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances. In Ingénierie des connaissances. Évolution Récentes et nouveaux défis. Paris: Eyrolles

[18] Uschold M. et Grüninger M., 1996. Ontologies: Principles, Methods and Applications. Journal of Knowledge Engineering Review. 11: 5-33.

[19] Corcho O., Fernandez-Lopez M. et Gomez P. A., 2002. OntoWeb Technical Roadmap, Version 1.0. http://www.ontoweb.org.

[20] National Committee for Information Technology Standards, Technical Committee T2 (Information Interchange and Interpretation). Draft proposed American national standard for Knowledge Interchange Format. http://logic.stanford.edu/kif/dpans.html, 1998.

[21] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua Server: a tool for collaborative ontology construction. In Proceedings of the 10th Knowledge Acquisition for KnowledgeBased Systems Workshop (KAW'96), 1996.

[22] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge Representation System. Cognitive Science, 9(2):171–216, April 1985.

[23] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, Peter PatelSchneider, editors. The Description Logic Handbook. Cambridge University Press, 2003;

[24] Dan Brickley and R. V. Guha, Editors. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, 10 February 2004.http://www.w3.org/TR/rdf-schema/

[25] D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: The very high idea. In Proceedings of the 11th International Flairs Conference (FLAIRS-98), Sanibal Island, Florida, May 1998.

[26] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. OIL: The Ontology Inference Layer. Technical

Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, Sept. 2000. http://www.ontoknowledge.org/oil/.

[27] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. DAML+OIL Reference Description. W3C Note 18 December 2001. http://www.w3.org/TR/daml+oil-reference.

[28] DARPA Agent Modelling Language-Ontology, Octobre 2000

[29] Deborah L. McGuinness and Frank van Harmelen, Editors. OWL Web Ontology Language Overview, W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-features

[30] Michael K. Smith, Chris Welty, and Deborah L. McGuinness, Editors, OWL Web Ontology Language Guide, W3C Recommendation, 10 February 2004. http://www.w3.org/TR/owl-guide/

[31] Paulo C. G. COSTA, Kathryn B. LASKEY. PR-OWL: A Framework for Probabilistic Ontologies The Volgenau School of Information Technology and Engineering, George Mason University, USA,2006

[32] Judea Pearl. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, 1988.

[33] P. Naïm, P-H. Wuillemin, P. Leray, O. Pourret, and A. Becker. *Réseaux bayésiens*. 423 pages, Eyrolles, Paris, 2007.

[34] Patrick Naïm, Pierre-Henri Wuillemin, Philippe Leray, Olivier Pourret, Anna Becker. Réseaux Bayésiens, Eyrolles 2008.

[35] Steffen L. Lauritzen et David J. Spiegelhalter. « Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems ». Journal of the Royal Statistical Society, Series B, 50 :157–224, 1988.

[36] Frédéric Santos. L'algorithme EM : une courte présentation, CNRS, UMR 5199 PACEA, 2015

[37] P. Naïm, A. Becker. Les reseaux bayesiens. 199 pages, Eyrolles, Paris, 1999

[38] Jean-Marc Trémeaux. Algorithmes génétiques pour l'identification structurelle des réseaux bayésiens, Rapport de stage de Master Recherche en Informatique Spécialité Extraction de Connaissances à partir des Données,Université Lumière - Lyon II, 2006

[39] Ross D. Shachter et Mark A. Peot. « Simulation Approaches to General Probabilistic Inference on Belief Networks ». Dans Proceedings of the Conference on Uncertainty in Artificial Intelligence, pages 221– 234, Amsterdam, The Netherlands, 1990.

[40] Robert Fung et Del B. Favero. « Backward Simulation in Bayesian Networks ». Dans Proceedings of the Conference on Uncertainty in Artificial Intelligence, pages 227–234, San Francisco, CA, USA, july 1994.

[41] Stuart Russell, Peter Norvig. Intelligence artificielle, 3eme édition, 1200 pages , 2010

[42] Paulo C. G. Costa & Kathryn B. Laskey, PR-OWL: A Bayesian extension to the OWL Ontology Language, 01/04/2016. http://www.pr-owl.org/mebn/index.php.

[43] Paulo C. G. Costa,Terry Janssen. Probabilistic Ontologies for Multi-INT Fusion. Kathryn Blackmond Laskey, Center of Excellence in C4 I George Mason University, USA.

[44] Zhongli Ding, Yun Peng, and Rong Pa. BayesOWL: Uncertainty Modelling in Semantic Web Ontologies, Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, Maryland 21250, USA

[45] Stefan Fenz. An ontology-based approach for constructing Bayesian networks, Vienna University of Technology, Favoritenstrasse 9-11/E188, 1040 Vienna, Austria, 2011

[46] Costa, PauloCesarG, KathrynB Laskey, and KennethJ Laskey PR-OWL : A Bayesian Ontology Language for the Semantic Web. In Uncertainty Reasoning for the Semantic Web I. P. Costa, C. d'Amato, N. Fanizzi, K. Laskey, K. Laskey, T. Lukasiewicz, M. Nickles, and M. Pool, eds. Pp. 88-107, 2008.

[47] Costa, Paulo CG. Bayesian semantics for the Semantic Web, George Mason University, 2005.

[48] Paulo C. G. Costa & Kathryn B. Laskey, PR-OWL: A Bayesian extension to the OWL Ontology Language, 01/04/2016. http://www.pr-owl.org/mebn/index.php.

[49] Olivier Francois, De l'identification de structure de réseaux bayésiens à la reconnaissance de formes à partir d'informations complètes ou incomplètes, THESE DE DOCTORAT, l'Institut National des Sciences Appliquées de Rouen ,2006.

[50] Laskey, Kathryn Blackmond, et al. 2011 PR-OWL 2 case study: A maritime domain probabilistic ontology. Proceedings of the 6th International Conference on Semantic Technologies for Intelligence, Defense, and Security, Fairfax, VA, 2011, pp. 76-83.

[51] Carvalho, Rommel Novaes. Probabilistic Ontology: Representation and Modeling Methodology, A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University, 2011.

[52] Donald G. Firesmith, Henderson-Sellers. The OPEN Process Framework: An Introduction, Pearson Education, 2002.