# Improving uncertain reasoning combining probabilistic relational models and expert knowledge

Mélanie Munch

Thèse de doctorat

NNT : 2020UPASB011

université PARIS-SACLAY

# Améliorer le raisonnement dans l'incertain en combinant les modèles relationnels probabilistes et la connaissance experte

**Thèse de doctorat de l'université Paris-Saclay**

École doctorale n° 581, Agriculture, Alimentation, Biologie, Environnement et Santé (ABIES)
Spécialité de doctorat: Informatique appliquée
Unité de recherche : Université Paris-Saclay, AgroParisTech, INRAE MIA-Paris, 75005, Paris, France
Référent : AgroParisTech

**Thèse présentée et soutenue à Paris-Saclay le 17/11 2020, par**

**Mélanie MUNCH**

## Composition du Jury

| | |
|---|---|
| **Mme Nacera SEGHOUANI BENNACER**<br>Professeure, CentraleSupélec | Présidente |
| **Mme Madalina CROITORU**<br>Professeure des Universités, Université de Montpellier | Rapporteur & Examinatrice |
| **M. Manfred JAEGER**<br>Professeur associé, Université d'Aalborg | Rapporteur & Examinateur |
| **M. Thomas GUYET**<br>Maître de conférences, l'Institut Agro – Agrocampus Ouest | Examinateur |
| **Mme Fatiha SAIS**<br>Professeure des Universités, Université Paris-Saclay | Examinatrice |
| **Mme Juliette DIBIE**<br>Professeure, AgroParisTech | Directrice de thèse |
| **Mme Cristina MANFREDOTTI**<br>Maître de conférences, AgroParisTech | Co-encadrante & Examinatrice |
| **M. Pierre-Henri WUILLEMIN**<br>Maître de conférences, Université Paris VI | Co-encadrant & Examinateur |

A ma famille, qui n'a jamais vraiment très bien compris ce que je faisais, mais qui m'a toujours soutenue pour que je le fasse.
*To my family, that have never quite understood what I was doing, but have always supported me while doing it.*
Et à papy, qui aurait fortement apprécié l'absence notable de souris dans ce manuscrit.
*And to grandpa, who would have appreciated the obvious lack of mouses in this manuscript.*

# ACKNOWLEDGEMENTS

Je n'aurais pu écrire ces lignes sans le soutien de nombreuses personnes, que je souhaiterai remercier ici.

Un immense merci donc tout d'abord à Juliette Dibie, Cristina Manfredotti et Pierre-Henri Wuillemin pour votre présence infaillible, vos conseils avisés et vos relectures impitoyables. Merci pour votre confiance, pour m'avoir soutenue et autant appris durant ces années. Merci pour votre expérience et vos critiques, qui m'ont permis de m'épanouir sur un sujet aussi intéressant. D'un point de vue scientifique comme humain, ce fut un véritable plaisir et honneur et travailler à vos côtés, que j'espère pouvoir retrouver dans ma vie professionnelle future.

Merci à l'unité MIA-Paris et au laboratoire EKINOCS - anciennement LINK - pour leur accueil dans les locaux d'AgroParisTech. En particulier, je tiens à remercier Liliane Bel, directrice de l'unité, et Antoine Cornuéjols, directeur de l'équipe, pour leur confiance et support. Merci également à Liliana Ibanescu et Christine Martin, sans qui Grignon n'aurait pas été aussi agréable; à Stéphane Dervaux, qui continue avec dévotion de manier sarcasmes caustiques et aide précieuse; à Joon Kwon, pour ses inestimables explications du dense *Causality*, de Pearl. Merci à Erica Helimihaja et Christelle Gehin pour leur patience lors la prise en charge des voyages et des billets d'avion, parfois bien compliqués. Enfin, merci à Pierre-Alexandre, Joe, Sema, Serife, Irène, Martina, Annarosa, Jules et Julie pour ces moments passés, qui à manger des burritos, qui à boire un chocolat, qui à donner des conseils en italien, qui à procrastiner en parlant de tout et de rien; cette thèse n'aurait pas eu la même saveur sans vous. Enfin, malgré un bref passage entravé par une clé capricieuse et une pandémie mondiale, merci au laboratoire LIP6 pour son accueil chaleureux, et pour m'avoir aidée à me faire une place.

Pour leur accompagnement au cours de ces trois ans et tous les perfectionnements qu'elles ont pu apporter, je tiens également à remercier Véronique Delcroix et Nathalie Pernelle pour leur implication dans le comité de suivi de thèse.

Un immense merci à Manfred Jaeger et Madalina Croitoru, qui ont accepté d'être rapporteurs pour cette thèse. Merci pour vos relectures et conseils. Merci à eux, et également à Thomas Guyet, Fatiha Sais et au Nacera Seghouani Bennacer pour avoir participé au jury lors de la soutenance, malgré les aléas de la visio-conférence. Merci pour les discussions, vos critiques et remarques qui vont m'aider à affiner ce travail.

Merci également aux élèves que j'ai pu voir défiler durant ces trois années, qui m'ont fait prendre conscience du plaisir que l'on peut avoir à enseigner. Oui, même avec toi, celui qui a préféré me passer le dictionnaire en *string* pour faire un *.found()* en Python plutôt que de faire un appel à dictionnaire comme c'était décrit en page 25 du polycopié de cours. Je ne t'ai pas oublié.

Merci aux organisateurs de la formation doctorale EIR-A d'Agreenium, pour les séminaires organisés, les rencontres effectuées et l'opportunité offerte de pouvoir effectuer un séjour à l'étranger.

Merci à l'équipe DISI du laboratoire d'informatique de l'université de Trento, pour leur accueil et pour m'avoir permis de découvrir de nouveaux domaines. En particulier, merci à Andrea Passerini et Paolo Dragone pour leur encadrement et leur accompagnement tous le long de ces trois mois.

Enfin, un immense merci pour l'encadrement offert par ABIES et notamment par Alexandre

Pery, toujours très à l'écoute des moindres problèmes. Vous apportez une aide précieuse pour de nombreux doctorants dans des moments parfois très pesants pour eux, et votre réactivité pendant une période compliquée comme celle de la pandémie a vraiment été d'un réconfort immense.

Le travail représente une grosse partie de la vie d'un doctorant. Néanmoins, celui-ci ne pourrait jamais être aussi bien mené sans le soutien indéfectible de la famille et des amis autour. Merci à mes parents, pour leur écoute et leur soutien; à mes frères et sœurs, et tout ceux qui ont pris le temps de m'envoyer du soutien pendant la thèse et la soutenance.

Beaucoup d'amis à remercier également: Hugo, Arthur, Fabien, Caroline, François, Antoine, Rémi, Rémi, Julien, Laure, Julien, Bérénice, Chloé, Axel... La liste pourrait continuer sur la longueur de la thèse, et plus encore. Mais je souhaiterai remercier en particulier Kévin, qui a toujours été là quand il s'agissait de m'écouter me plaindre, et a toujours su donner les conseils dont j'avais besoin.

Merci à Paul-Henri, qui a toujours été là, même lorsque j'étais à l'autre bout du monde; qui n'a jamais douté, et m'a toujours soutenue.

Et enfin, merci à Ainu et Arda, dont le talent n'équivaut pas à celui de F. D. C. Willard, mais qui ont au moins eu le mérite d'avoir offert un joyeux divertissement de fin de soutenance.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

'I'm sure. But look at it this way. What really is the point of trying to teach anything to anybody?'

*This question seemed to provoke a murmur of sympathetic approval from up and down the table. Richard continued.* 'What I mean is that if you really want to understand something, the best way is to try and explain it to someone else. That forces you to sort it out in your own mind. And the more slow and dim-witted your pupil, the more you have to break things down into more and more simple ideas. And that's really the essence of programming. By the time you've sorted out a complicated idea into little steps that even a stupid machine can deal with, you've certainly learned something about it yourself. The teacher usually learns more than the pupil. Isn't that true?'

- Douglas Adams, **Dirk Gently's Holistic Detective Agency**

# INTRODUCTION

The idea of an artificial intelligence (AI) able to define concepts and reasoning with them has always been an appealing idea, even at the earliest beginning of computer science. Widely considered as the first computer programmer, Ada Lovelace acknowledged in 1843 in her notes describing the first algorithm in history (dedicated to compute Bernoulli numbers) that the true potential of computers was their ability to deal with abstract concepts (Menabrea et al., 1843) rather than just being number crunchers. Although she also conceded that at that time machines were only capable of computation and not creation, the concept of an AI able to apprehend and mimic complex human reasoning kept being brought back, in particular by Alan Turing who even designed a test to evaluate the potential of such an AI (the famous Turing test, still never succeeded to this day) (Turing, 1950).

In the mid 1970s the field of **expert systems** emerged with the goal of mimicking the decision making process of a human, usually narrowed to a specific domain, such as medicine with MYCIN (van Melle, 1978), an expert system dedicated to diagnose infectious diseases, or chemistry with DENDRAL (Buchanan and Sutherland, 1968), able to infer molecules structure from spectroscopic analyses. Expert systems are based on the transcription of *expert rules* (such as: "*If* a customer buys a car, *then* they might be interested in an insurance") into logical predicates that allow logical and probabilistic inferences. Expert systems are divided into two distinct parts: the **knowledge base**, that encompasses all the facts and rules the system needs to know; and the **inference engine**, that combines the rules of the known facts to infer new truths. If at the beginning the knowledge bases were fairly small, they quickly grew in size with time. As a consequence the problem of their structuring was raised, and how it should be dealt with in order to ease the "human to computer" translation. This gave room to **knowledge engineering**, term coined by Freigenbaum(Feigenbaum and McCorduck, 1983), a key contributor to DENDRAL. The main

idea behind this new domain was to give tools to build, maintain and interact with resources previously defined by human operators. The aspect of knowledge representation was here a key feature, to which **ontologies** brought an answer.

In the 1980s ontologies were considered in AI both as a philosophical concept to describe the world, and as a component for knowledge-based systems. The first use of the term in the way we now know it can be traced to Tom Gruber, who laid the foundations of their current definition. In his work, he defines ontologies as a group of domain's concepts and relations that can be used to describe any agent or community of agents (Gruber, 1995). Indeed, ontologies organize and structure the knowledge in terms of concepts, relations between them, and instances (Staab and Studer, 2009). According to Feilmayr and Wöß, ontologies' success stems from this huge adaptability: "An ontology is a formal, explicit specification of a shared conceptualization that is characterized by high semantic expressiveness required for increased complexity."(Feilmayr and Wöß, 2016). This adaptability allows them to be defined on different levels depending on their genericity (Guarino, 1998). In this thesis, we will focus on **domain ontologies**, which are used as a common and standardized vocabulary for representing any domain (e.g. life-science, geography). Those ontologies can be defined using different reference languages as recommended by the W3C[1]. In the following, we will consider only the RDF[2] and OWL[3] languages, which are based on the XML markup language.

**RDF** is a formal representation model for resources where data is written as <subject, predicate, object> triple. Subject and predicate are denoted by a unique Universal Ressource Identifier (URI); in the case of the object, it can also be a unique URI, or a datatype (such as a number, a name, etc.), depending of the relation we want to describe. For instance:

- < **db:International_Semantic_Web_Conference**, **dct:subject**, **dbc:Artificial_intelligence_conferences**> [4] identifies the subject of the International Semantic Web Conference (ISWC) as the AI, which is also another resource of this website, with other predicates.
- <**db:International_Semantic_Web_Conference**, **dbp:history**, "2002">, on the contrary, states that the ISWC has been created in 2002, which is strictly a value.

**OWL** helps to extend the RDF notation by adding the concepts of classes, properties and axioms, which provide more semantic. In this thesis, we will mainly use the concepts of:

- **Class:** a class represents a *entity* such as a conference, a movie, an animal... Semantically it

[1] World Wide Web Consortium, who promotes standards for every web-related developments: http://www.w3.org/XML
[2] https://www.w3.org/RDF/
[3] https://www.w3.org/OWL/
[4] For a better readability those URI have been shortened using a prefix. *db* stands for *http://dbpedia.org/page/*, *dct* for *https://dublincore.org/specifications/dublin-core/dcmi-terms/subject#*, *dbc* for *http://dbpedia.org/page/Category* and *dbp* for *http://dbpedia.org/page/property*

embodies the same entity (e.g. a lion) that can then be instantiated multiple times. **Example:** *zoo_1*, which is an instance of the class ***Zoo***, has three lions, instances of the class ***Lion***: *lion_1*, *lion_2* and *lion_3*.

- **Object Property:** an object property is a relation that links two classes or instances, with one as the subject (the domain) and one as the object (the range). This relation is written as a triple. **Example:** the property ***hasForAnimal*** helps to define the animals of a zoo: <*zoo_1*, ***hasForAnimal***, *lion_1*>. A symmetric property can also exist: <*lion_1*, ***isAnimalOf***, *zoo_1*>.
- **Datatype Property:** a datatype property is a relation that links a class or an instance to a data value. The individual and the data are always respectively the subject and the object. **Example:** the property ***hasForAge*** attributes an age to each animal, for instance <*lion_1*, ***hasForAge***, "15">. Since the object is always a data value, there is no symmetric property for the datatype properties.

Thanks to the introduction of ontologies, domains can now easily be structured. However, the more complex they grow, the more difficult their data's analysis also becomes. Indeed, adding more facts opens to more issues, such as incomplete data, which renders analysing more challenging since information is missing. In addition to this, there is intrinsic uncertainty in many complex domains, as the observed phenomena are not always fixed and cannot be determined with certainty.

> **Example 1.** An omniscient robot would be able to give an accurate weather estimate of tomorrow, as it would know every relevant parameters to compute its prediction. A common human, however, would only have access for instance to the sky; the only prediction they would be able to give would be of the form *"If the sky is cloudy, then rain is highly probable"*.

All of these new issues add subtleties that need specific tools in order to study and analyze them. Since ontologies and classical system experts are dedicated to logical reasoning, other various approaches have been proposed to expand their ability to work with such complex domains, such as for instance probabilistic reasoning which is best suited for dealing with uncertainty. Frameworks have already been proposed:

- **BayesOWL** (Ding et al., 2006, Zhang et al., 2009) and a similar framework **OntoBayes** (Yi Yang and Calmet, 2005) directly represent a probabilistic model using the OWL notation.
- **MEBN/PR-OWL** and **PR-OWL 2.0** (Carvalho et al., 2013, da Costa et al., 2008) represents in a similar fashion a probabilistic model using PR-OWL 2, an upper ontology written in OWL.
- **HyProb-Ontology** (Mohammed, 2016) presents a hybrid ontology able to deal both with

continuous and discrete data in an ontology (on the contrary of the previous frameworks that were only able to deal with discrete variables).

These frameworks represent an interesting and efficient way of modeling the probabilistic dependencies, while offering a way for using probabilistic inference tools within the ontology. However they do not offer a way to learn such dependencies between the different values of the ontology.

In this thesis, we present novel methods in order to bridge this gap between ontological knowledge and probabilistic reasoning. Our work will present and develop different methods to exploit and reason on the knowledge encapsulated in knowledge bases using probabilistic learning.

#### OBJECTIVES

Reasoning over a domain requires to have the objects of interest modeled, usually through the means of different variables. In Example 1, for instance, the weather can be represented by the rain frequency, the clouds opacity, etc. Therefore, a good modeling of such a domain requires to represent these variables. However, if we want this model to be explanatory, we need to represent the probabilistic dependencies between these variables. To do that, the scientific method for instance rests on several well-defined steps, with among them formulating and verifying hypothesis through experiments. Experiments, however, are sometimes difficult to carry out, for legal, practical, temporal, economical or ethical reasons: to this intent a model can help amplify the human thought process (Churchman, 1968), and give scientists a better overview on the whole operation. As a consequence, modeling is a large part of the scientific method: it helps to encompass all known facts, to reason with them as a whole set, and to check the formulated hypothesis for making predictions.

Yet learning an accurate modeling of any given domain can also be a hard problem, which nearly every time requires a good grasp of its issues. Indeed, the choice of the model basis and its key components is essential and, if done wrongly, can throw off the whole model's predictions. For instance in statistical models, the most critical part of the analysis usually depends of the quality of the translation from the core subject to the model (Cox, 2006).

In this thesis, we offer new methods to use knowledge about domains encompassed in **knowledges bases** in order to improve probabilistic reasoning. The term "Knowledge base" is often used interchangeably in literature with other such as *"ontologies"* or *"knowledge graphs"* (Ehrlinger and Wöß, 2016). Since Google introduced this last term in 2012 to refer to the use of semantic knowledge in Web search, no concrete definition has been given. Several attempts have been

made on the requirements a knowledge graph should have, based on characteristics such as the size (Huang et al., 2017), or on the restriction only to RDFs bases (Färber et al., 2017). In this dissertation, we will consider the definition of Ehrlinger and Wöß (2016) in which a knowledge base is an ontology structure combined with a large population. More formally, we denote a knowledge base $\mathcal{KB}$ as a couple ($\mathcal{O}$, $\mathcal{F}$) where the ontology $\mathcal{O}$ is represented in OWL and the knowledge graph $\mathcal{F}$ represents the data in RDF.

- The ontology $\mathcal{O} = (\mathcal{C}, DP, OP, A)$ is defined by a set of classes $\mathcal{C}$, a set of ***owl:DatatypeProperty*** $DP$ in $\mathcal{C} \times T_D$ with $T_D$ being a set of primitive datatypes (e.g. integer, string), a set of ***owl:ObjectProperty*** $OP$ in $\mathcal{C} \times \mathcal{C}$, and a set of axioms $A$ (e.g. property's domains and ranges).
- The knowledge graph $\mathcal{F}$ is a collection of triples $(s, p, o)$, called instances, where $s$ is the subject of the triple, $p$ a property that belongs to $DP \cup OP$ and $o$ the object of the triple.

**Our main goal is to learn a probabilistic model in order to discover new knowledge using the $\mathcal{F}$ triples and the semantic information brought by $\mathcal{O}$.** We chose to focus on specific domains that shared common features such as uncertainty, missing values, complex relations between the potential variables that were possible to model using probabilistic models. Probabilistic models are dedicated to represent random variables and the relations between them under the form of probabilistic distributions (that we will detail in Chapter 1). This allows (1) a good flexibility, as all domains with uncertainty can be represented and (2) a good readability. Indeed, on the contrary of some "black-box" models such as the neural networks, that present a very efficient way to learn but a poor explainability of the learned results, probabilistic models allow the analysis of the relations between the variables. This usually gives a visual and interpretable model, useful when dealing with complex domains.

In this thesis we present and develop three different methods for guiding the learning of a probabilistic model using expert knowledge. The novelty of our approach is that this expert knowledge is brought from the knowledge base on one hand, and from a human expert of the domain on the other hand, which guaranties a model learned as close as possible to the target domain. Moreover, throughout our work, one of our main goals was the accessibility for the so-called experts. This accessibility is expressed on two levels: (1) during the extraction of the ontological knowledge used for the learning, and (2) for the learned model exploitation afterwards. For all our contributions the focus has been set notably on causality as part of causal discovery, which is an hard but rewarding goal when it comes to modeling and understanding complex domains.

The originality of our approach is the combination of two research domains with broadly

different paradigms. Both are dedicated to describe a model as close as possible to the reality, but with two different visions: while the Bayesian approach favors the statistical analysis of the given data in order to discover knowledge among what it already knows, the ontological approach is based more on the information brought by the expert in order to discover new explicative and predictive rules. In these thesis however we will demonstrate how both of these approaches can mutually benefit by 1) enhancing the reasoning with probabilistic variables by introducing semantic knowledge and 2) enriching the ontological knowledge by learning new rules.

### Thesis Outline and Contribution

This thesis is organized in several chapters. All of the original works are validated and illustrated with complete examples that were used to validate our publications.

- **Chapter 1. Background and State of the art** gives an overview on the different methods we used and the state of the art. It focuses on probabilistic models such as Bayesian networks (BNs) and Probabilistic Relational Model (PRMs), as well as their learning, especially under constraints. A second part of this chapter is dedicated to causality, causal discovery, and the main challenges they raise.
- **Chapter 2. Learning a Probabilistic Relational Model from a specific ontology** presents an example drawn from a given ontology, and how it can be used to learn a probabilistic model. In this chapter, we focus on how using an ontology can greatly help to learn a model closer to the reality. We introduce here our first contribution, the **ON2PRM algorithm**, which allows to learn a probabilistic relational model from any domain ontology using the specific core ontology presented in this chapter.

  **Publications.**

  - **Munch M.**, Wuillemin PH., Manfredotti C., Dibie J. and Dervaux S. Learning Probabilistic Relational Models using an Ontology of Transformation Processes. *In:* ODBASE 2017.
- **Chapter 3. Interactive learning from any knowledge base** presents **CAROLL**, a first algorithm dedicated to build a relational schema from any knowledge base by using a human expert contribution. This method is guided by a causal assumption formulated by the expert. The aim of the causal assumption here is to motivate the learning of the model by giving it a precise purpose: checking whether the expert's belief is true or not. In this chapter, we present this method as well as the causal discovery aspect of our work.

  **Publications.**

  - **Munch M.**, Wuillemin PH., Dibie J., Manfredotti C., Allard T., Buchin S. and Guichard

E. Identifying control Parameters in Cheese Fabrication Process Using Precedence Constraints. *In:* DS 2018.

  - **Munch M.**, Dibie J., Wuillemin PH. and Manfredotti C. Towards Interactive Causal Relation Discovery Driven by an Ontology *In:* FLAIRS 2019.

- **Chapter 4. Semi-automatic learning from any knowledge base** presents **ACROSS**, an algorithm where the contribution required from the expert is reduced compared to the algorithm introduced in the previous chapter. Our aim here is to be able to learn a probabilistic model from any knowledge base while easing as much as possible the workload asked to the expert. In this chapter we give a detailed overview of the differences with the previous methods, as well as the tools given to the expert in order to help them.

  **Publications.**

  - **Munch M.**, Dibie J., Wuillemin PH. and Manfredotti C. Interactive Causal Discovery in Knowledge Graphs. *In:* PROFILES/SEMEX@ISWC 2019.

- **Conclusion and Perspectives** summarizes the results presented in these thesis, discuss their limitations and presents some perspectives works for the future.

**Domain studied in this thesis**

Several in-use knowledge bases will be presented in this thesis, reflecting the adaptability of our methods. One uses the Process and Observation Ontology[a] ($PO^2$), dedicated to transformation process; and two have been extracted from DBPedia[b], restricted to the subjects we wanted to analyze. The first is about movies[c] and have been enriched with information from the website ImDB[d]. The second is about books' authors[e].

[a] http://agroportal.lirmm.fr/ontologies/PO2
[b] https://wiki.dbpedia.org/
[c] https://bit.ly/2RYVjG8
[d] http://www.imdb.com/
[e] https://bit.ly/2X0eeCw

# CHAPTER 1

# BACKGROUND AND STATE OF THE ART

The main challenge of our approach is the combination of two different domains of artificial intelligence that are not usually studied together: knowledge representation and uncertainty reasoning. We already have briefly presented, in the introduction, knowledge bases and how we define them in this thesis. In order to better understand the issues and challenges raised, this chapter presents the state of the art on the different tools we have used.

**Section 1.1** describes probabilistic theory (1.1.1), and more especially Bayesian networks (1.1.2), how to learn them (1.1.3), and essential graphs (1.1.4). In subsection 1.1.5, we present a Bayesian networks' object-oriented extension: the probabilistic relational models. The goal of this thesis is to propose an approach combining ontologies and probabilistic models in order to learn a model semantically close to the reality. As a consequence we do not offer an extended comparison between the different learning methods, as it falls out of the scope of our study. That is why in these sections we only briefly touch the learning algorithms and scores chosen for the rest of our work; but since learning using ontology knowledge is similar to learning under constraints, we develop the current state of the art on the matter (1.1.6). The last section is dedicated to explain the works aiming to combine probabilistic models and ontologies (1.1.7).

**Section 1.2** presents the main ideas we develop in this thesis on the matter of causality. It first covers the principal definitions and tools described in the literature (2.2.1) that we use in this thesis. Afterwards it presents the problem of causal discovery (1.2.2) and how it is handled in other works. It concludes with some thoughts on causality, ontologies and explanation (1.2.3).

## 1.1 Probabilistic Models

Probabilistic models are an efficient way to express probabilistic dependencies between different variables. They fall into different categories, depending on the way these relations are expressed. In these thesis, we will concentrate on probabilistic graphical models which use graph theory to show the conditional dependence structures. In order to introduce these models, we first give a short review on discrete probability theory.

### 1.1.1 Discrete Probability Theory

We define a **random variable** as a variable that can present different states. For each problem we wish to study, we define a set of variables able to describe all of its parameters.

> **Example 2.** We define three random Boolean variables $R$, $S$ and $G$. These respectively accounts for whether it is raining (*True*) or not, whether the sprinkler is on (*True*) or not, and whether the grass is wet (*True*) or not. Our problem is then described by the Cartesian product of all the possible values for each variable: { ($R$,$S$,$G$), ($\neg R$,$S$,$G$), ($\neg R$, $\neg S$,$G$)...}.

The goal of **discrete probability theory** is (1) to evaluate, for each of the defined variables, the mapped probability function; and (2) to express, if possible, the impact of a group of variables over the others. To do so, we denote $X$ as a random variable, and $P(X)$ its associated law.

Using it, discrete probability theory aims to answer two kinds of questions:

1. "What is the probability that $X$ takes the value $x_i$ and $Y$ the value $y_i$?", which is denoted $P(X = x_i, Y = y_i)$. $P(X, Y)$ is known as the **joint probability**.
2. "What is the probability that $X$ takes the value $x_i$ knowing that $Y$ **has taken** the value $y_i$?", which is denoted $P(X = x_i|Y = y_i)$. $P(X|Y)$ is known as the **conditional probability** and is read "probability of $X$ knowing $Y$". It is important to note that this probability is defined only if we deal with a non-zero intersection, i.e. $P(Y = y_j) > 0$.

**Independence**

In the particular case where $X$ and $Y$ are **independent** (meaning the value of one has no influence over the value of the other), those quantities can easily be computed, with:

1. $P(X = x_i, Y = y_i) = P(X = x_i)P(Y = y_i)$
2. $P(X = x_i|Y = y_i) = P(X = x_i)$

We distinguish discrete from continuous probability theory with the types of random variables used to describe the problem. Indeed, if the number of variable's states is finite or countable, then

we consider the variable as **discrete**, meaning all of its states can be singled out. On the contrary, if the states are represented by the set of real numbers $\mathbb{R}$ (or a portion of it), then the variable is considered as **continuous**.

**Example 3.** When we consider a coin throw, the space is usually finite, with each state referring to a possible outcome: $D_{coin} = \{\mathsf{head}, \mathsf{tail}\}$. On another hand, when we consider the average income of a population, we deal with quantities that cannot realistically be listed, thus: $D_{income} = \mathbb{R}^+$. If we want to study it with the discrete probability theory, we thus have to discretize its values, by creating categories in which every possible value can be sorted.

**Determining ($X_1, X_2, ...X_n$)**

Defining a space sometimes requires to **discretize** all the possible outcomes, event the less likely.

The discretization occurs when we want to specify different categories among those we are presented with. In this thesis, the most usual case is when we have a continuous space that we want to transform into a discrete one: for instance, $D_{income} = \mathbb{R}^+$ can become $D_{income} = \{D_{<1200}, D_{\geqslant 1200}\}$ where we distinguish between when the income is less than 1200 and when it is bigger.

If we consider a coin's tossing, a lot more is theoretically possible than just head or tail: the coin can fall on its edge, or be lost, or stolen,... We however chose to consider that the coin will always fall back either on head or tail, and ruled out these other possibilities.

In this thesis, we will only briefly cover this subject, as it is not at the core of our work. It is however essential to keep in mind that **defining the different states we will take into account in our problem is already taking a stance in the modeling of our event**

Both discrete and continuous variables have different properties and definitions; in the following we will only consider the discrete probability theory. This implies that all the variables we deal with are either (1) discrete or (2) have been discretized. Discrete probability theory attributes to each variable's state a probability that varies between 0 (the state will never occur) and 1 (the state is certain); the probability of the union of all the possible states is 1, meaning the space represents indeed all of the possible values. For the following, we denote $P(x)$ the **marginal probability** representing $P(X = x)$.

**Example 4.** If in our coin example the coin is balanced (meaning one side is not favored over the other), then we have $P(\mathsf{tail}) = P(\mathsf{head}) = 0.5$: no event is more likely to happen than the other. Moreover, $P(\mathsf{tail}) + P(\mathsf{head}) = 1$, meaning that we do not take into account other

outcomes.

In a more general setting, we can express the **rule of the marginal probabilities sum** and the **rule of the probabilities product**. Be $\mathsf{Domain}_V$ the set of all possible values of the random variable $V$, we have

**Property 1.** $\forall x \in \mathsf{Domain}_X, \forall y \in \mathsf{Domain}_Y, P(x,y) = P(x|y)P(y) = P(y|x)P(x)$

It can be written as $P(X,Y) = P(X|Y)P(Y)$.

**Property 2.** $P(x) = \sum_{y \in \mathsf{Domain}_Y} P(x|y)P(y)$.

It can be written as $P(X) = \sum_Y P(X|Y)P(Y)$.

Using Property 1, Thomas Bayes defined in 1763 his famous Bayes' rule (Bayes, 1763).

**Theorem 1:** *Bayes' rule*. Being $X$ and $Y$ two random variables such that $P(Y) \neq 0$, we have

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

This theorem helps to express the *a posteriori* probability of $X$, meaning the probability of $X$ after $Y$ has been observed. This is especially useful as it helps to compute any probability, given we know (1) its conditional probabilities and (2) the marginal laws *a priori*. We can also express Bayes' rule as $P(X|Y) \propto P(Y|X)P(X)$, meaning that $P(X|Y)$ (the *posteriori*) is proportional to the product of $P(Y|X)$ (the *likelihood*) and $P(X)$ (the *a priori*). Following this intuition, Pearl (1988) presented a probabilistic graphical model based on Bayes' rule, the Bayesian network.

### 1.1.2 BAYESIAN NETWORKS

A Bayesian network is a probabilistic graphical model based on the representation of different variables and their influence on each other using tools from the graph theory.

**Definition 1:** *Bayesian Network*. A Bayesian Network of dimension $n$ is defined by:

- a *directed acyclic graph* G = (V,E), where V and E are respectively the set of all its nodes and arcs. V contains $n$ nodes, each representing a variable. For simplification's purpose, we associate in the following the variables with their graphical representation (i.e. their representing nodes).
- a *set of random variables* equal to V and defined such that:

$$p(V) = \prod_{x \in V} p(x|Parents(x))$$

where $Parents(x)$ is the set of all parent nodes of $x$ in the graph.

Since the value of a variable depends only on the values of its parents (i.e. the parents of the associated node in the graph), then a Bayesian network is the graphical representation of the dependencies between the variables. To each node with parents, a **conditional probability table** is assigned, which gives the probability values of each value of the variable in function of the value of its parents variables.

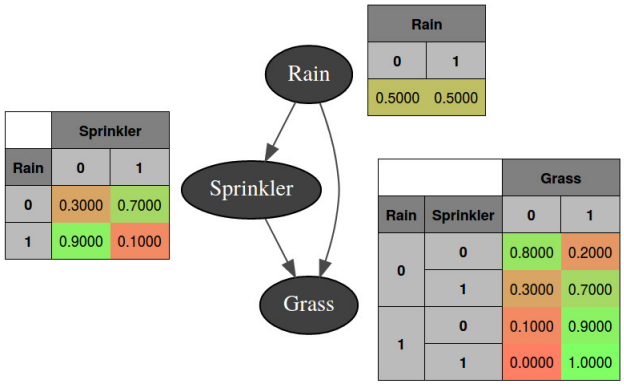



***Figure 1.1:*** **Example of a Bayesian Network using Example 2.** Each variable has a conditional probability table that shows how their values depend on their parents.

This allows a double reading very useful for our analysis. Suppose we want to model the system described in Example 2, and that we have learned the Bayesian network presented in Fig.1.1. We can then easily answer two types of questions:

- **Qualitative questions:** *"Does the value of the grass variable depend on the value of the rain variable?"* A simple look at the graph can answer: since there is an arc from the *Rain* node to the *Grass* node, then we can deduce they are not independent. We can also see that the *Rain* is not the only variable with an influence over the *Grass*, since *Sprinkler* and *Grass* also share an arc.
- **Quantitative questions:** *"What is the influence of the rain variable over the value of the sprinkler variable?"* We can then look at the conditional probabilities table and answer: for instance, if we have *Rain=True*, then the probability that the *Sprinkler* is actually On is 0.1, which is low. However, if in this example the answer can appear as trivial (since a simple look at the graph could help answer it), these kind of questions can become much harder when involving non-direct relations (such as: *"What is the influence of the rain variable over the value of the grass variable?"*).

In order to allow this double analysis, we first need to learn the Bayesian network. It requires two steps: a first for **learning the structure** of the graph, and a second to **learn the probabilistic parameters**.

### 1.1.3 LEARNING BAYESIAN NETWORKS

Learning a Bayesian network's structure is a NP-hard problem (Chickering et al., 2002), since as the number of variables goes up, the number of possible structures also raises: a 10 variables problem has for instance approximately $4.10^{18}$ possible solutions. As a consequence, the most common way to efficiently learn a structure is to look for graphs subsets in order to eliminate the least effective solutions and avoid considerable amount of useless calculations. There are two principal manners for selecting these structures: independence (or constraints) based algorithms, and score based algorithms .

- **Constraints based algorithms** (e.g. PC Spirtes et al. (2000)) are based on statistical tests. They are structured in three steps. Firstly, a non-oriented graph is built using independence tests between the variables. Then, once the first graph has been built, other independence tests are used to single out the specific structure $A \rightarrow B \leftarrow C$, also called **V-structure**. Indeed, V-structures indicate a complete independence between $A$ and $C$, which can help orient the edges (an example on a possible use of V-structures to orient edges is given in Example 19). Finally, once all V-structures have been identified, the rest of the edges are also oriented, being careful not to create V-structures.
- **Score based algorithms**, on the contrary, are composed of two parts: a search algorithm (e.g. Greedy Equivalent Search GES Chickering (2003)) and an heuristic score (e.g. AIC (Akaike, 1974), BIC (Schwarz, 1978) or BDeu (Buntine, 1991)). At each iteration of the algorithm, a change is brought to the graph (addition, removal, inversion of an arc) and a new score is computed. Scores are composed of two parts: firstly, they tend to search for the best structure that maximizes it. However, this component alone is not good enough, as it tends to favor complete graphs (i.e. graphs where each node are linked to the others), which usually result from over-fitting. As a consequence, scores also have to take into account a simplification factor, which applies Occam's razor philosophy: the simplest solution is usually the best.

From an historic point of view, constraint based algorithms (such as PC) learning results have been considered as better than score-based ones, due to the use of statistical tests. Moreover, in the scope of causal analysis theory that have been defined over the past years, these methods are also better than the score based algorithms. Yet, the statistical tests constitute also one of their weaknesses, as they can become quickly very demanding of data in order for their predictions to be robust. That is why score based algorithms are in the end usually more used than constraint based.

The main idea of our work is to integrate ontological knowledge as a constraint into the learning. However, the way classical constraint based algorithms are defined do not allow knowledge other than the one deduced from independence tests. In particular, even if we try to integrate our own *semantic* constraints, conflicts can be raised if they contradict what the algorithm have determined (e.g. two variables are statiscally dependent but the ontology tells us that they are not). As a consequence, we chose to use an heuristic algorithm, as they were the most efficient solution for integrating constraints from ontological and expert knowledge.

Once a structure has been found the probabilistic dependencies can be learn. To do so, two methods are also possible: a statistical learning based on the maximization of likelihood, or a Bayesian learning based on the estimation of the parameters considering the data set as additional unobserved variables. This last method, however, is very demanding and expensive, making the first solution easier to use. The learning of parameters consists thus in estimating $P(X|parents(X))$ in the dataset. This is mainly done by estimation of the frequencies and the consideration of *a priori* (Koller and Friedman, 2009).

One of the drawbacks of this approach is that when the scores are very close a structure can be chosen over another although both are valid choices. In particular, this can be highly critical when it comes to arc orientation. As stated before, the orientation of an arc in the Bayesian network only shows that one variable's values depend on another variable's values, given the computed probability tables. However, whether the orientation would be $A \rightarrow B$ or $A \leftarrow B$ is usually not significant, as the marginal probabilities $P(A)$ and $P(B)$ are the same: those structures are said to be part of the same Markov equivalence's class.

**Method used**

For the rest of this thesis, unless otherwise stated, we learn Bayesian networks using a Greedy Hill Climbing algorithm with the BIC score.

### 1.1.4 Essential Graphs

Learning a Bayesian network requires two steps: learning the structure, and then learning the probabilistic parameters from data on that structure. However, even if two structures are different, they can sometimes lead to the same joint distributions, as shown in Fig.**??**.

On the contrary, some arcs' orientation cannot be reversed without modifying the probabilistic dependencies. This is the case when they indicate independence between the nodes.

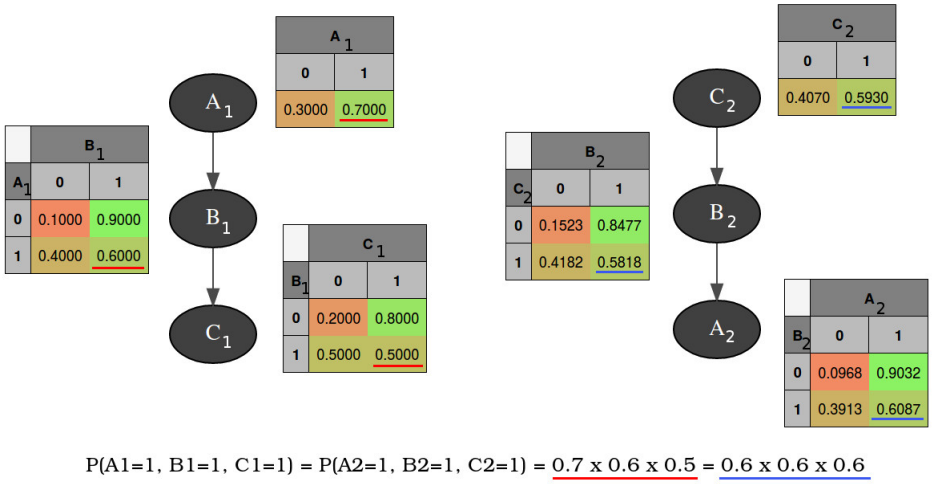



***Figure 1.2:*** **Bayesian Networks's equivalence.** In this example, $BN_1$ and $BN_2$ have a different structure but lead to the very same joint distributions. They are said to belong to the same Markov equivalence's class.

**Example 5.** Given three variables $A$, $B$, $C$, a structure such as $A \rightarrow B \leftarrow C$ indicates an independence between $A$ and $B$: fixing one (i.e. imposing its value) can give information about the other, but will have no impact on it. This particular configuration is called a **V-structure**, and its arc's orientation cannot be reversed without changing the nodes' independences: if we reverse $A \rightarrow B$ or $B \leftarrow C$, then $A$ and $B$ are no more independent.

**Definition 2:** *Immorality*. An **immorality** is a V-structure $A \rightarrow B \leftarrow C$ where there is no path linking $A$ and $C$.

An essential graph (Madigan et al., 1996) is a semi-oriented graph designed to represent immoralities in graphs. Every Bayesian network has one, with which it shares the same skeleton (i.e. the same graph structure but non-oriented). It has two kind of edges:

- **Oriented arcs** which designate all the Bayesian networks' arcs that are oriented the same way. Indeed, if all the models that have the same equivalence class have an arc oriented the same way, then this orientation is kept in the essential graph.
- **Non-oriented arcs** which, on the contrary, designate Bayesian networks' arcs that can be reversed without modifying the probabilistic dependencies. They are called **non-essential arcs**.

Two different Bayesian networks can share the same essential graph. In this case, they are said to be part of the same Markov equivalence's class. Essential Graphs are used to find arcs' orientation that are dependent of the learning: if an arc is oriented in the essential graph, then it means that no matter the learning, it will always be that way. They are not causal by essence, but under the right hypothesis (that we will detail later), they can be used to deduce causal knowledge.

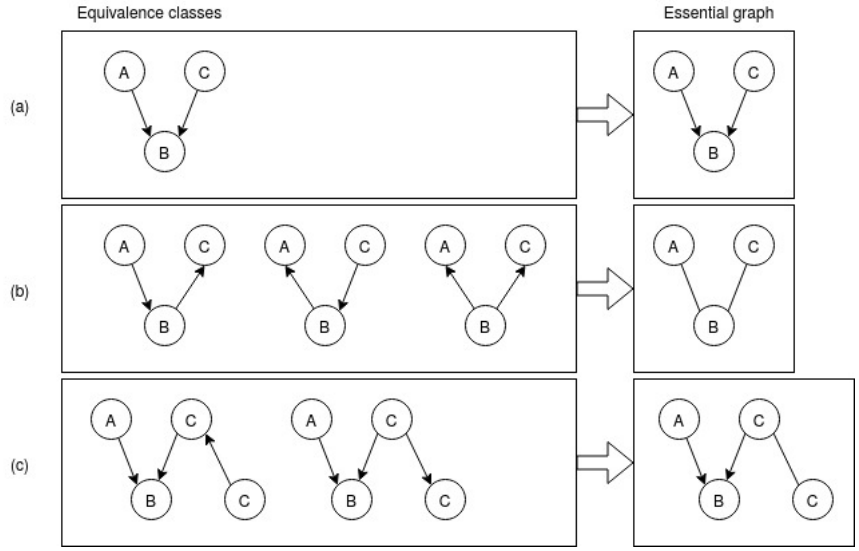



***Figure 1.3:*** **Example of equivalence classes and their associated essential graph.** (a) In this example, due to the V-structure, there is only one Bayesian network in the equivalence class, which thus has the same structure as the essential graph. (b) In this example, the equivalence class is composed of three Bayesian networks. Since none of the arcs are oriented the same way in the three examples, the essential graph is only composed of edges. (c) In this example, the equivalence class is composed of two Bayesian networks sharing a V-structure. The resulting essential graph is composed here of oriented and non-oriented edges.

### 1.1.5 PROBABILISTIC RELATIONAL MODELS

Bayesian networks are useful to represent probabilistic dependencies between a set of nodes. Their learning only requires a database composed of distinct examples with all the variables and their values. However, they can be limited when presented with complex settings with multiple nodes as they make no distinction between those during the learning. This is the case whenever one wants to learn a probabilistic model where a same group of nodes is repeated multiple times (see Example 6 for a detailed example). Since the Bayesian network cannot make distinctions, then this group has to be learned as many times as it is present, which increases the error margin.

**Example 6.** Be a process requiring the use of an oven for multiple steps. This oven is represented by three attributes: the temperature $Te$, the time $Ti$ and the energy $E$. The higher and longer the temperature and time are, the higher the consumed energy is: the probabilistic relation is $Te \rightarrow E \leftarrow Ti$. Since the oven is used at multiple steps during the process, then we have multiple measurements for the oven's attributes, one for each use, that we denote $Step_N.X$ for *"attribute X measured at Step N"*. As a consequence the learning set is composed of multiple instances of the same attributes measured at different times, as shown in Fig.1.4. Our goal would be to learn $Te \rightarrow E \leftarrow Ti$; however, this is not possible, as the learning makes no link between $Step_i.X$ and $Step_j.X$, which are considered as two different attributes. More-

over, attachment to a class is not taken into account: $Step_i$ and $Step_j$ are not considered as two different entities. As a consequence, in the best case scenario, we would only be able to learn two small networks $Step_1.Te \rightarrow Step_1.E \leftarrow Step_1.Ti$ and $Step_N.Te \rightarrow Step_N.E \leftarrow Step_N.Ti$, instead of the only one we would like to learn.

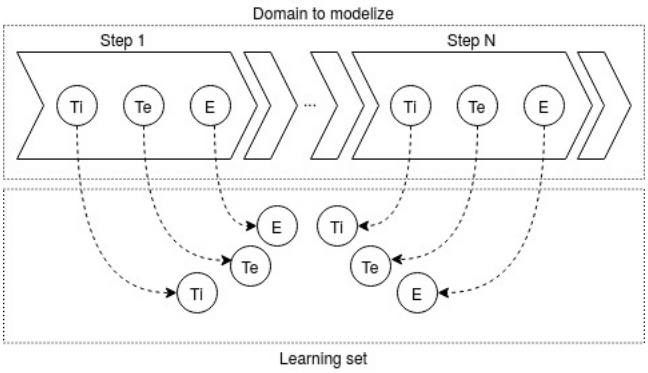



***Figure 1.4:*** **Modeling of a simple process.** If we want to model a process using multiple times an oven represented by three attributes $Te$, $Ti$ and $E$, then the database used for the Bayesian network will have multiple instances of these attributes with no way of distinguish them as "part of the same entity".

This is particularly restraining for the modeling of complex domains described by ontologies, where a same class can be instantiated multiple times. Ideally, we would like our learning to take into account the structuring brought by the ontology, where each concept is represented by an OWL class that can be instantiated one or multiple times. Considering this would require the learning to be able to comprehend:

- **Object properties.** They organize the relations between the classes that should be exploited. For instance, a *isBefore* object property creates a temporal property between two classes. Given two instants $t_i$ and $t_j$ such that $t_i < t_j$, an attribute from $t_i$ can be parent of an attribute from $t_j$, but not the contrary (since we want to learn an explanatory model the future cannot have an influence on the past).
- **Classes instantiations.** If a same concept (i.e. an apparel, a person, ...) intervenes multiple times in our domain, we would like to learn a single model to represent it, and not one model for each instance of this concept. This is the case developed in Example 6.

This cannot easily be done with a classical Bayesian network, and that is why in this thesis we propose to couple ontologies with an oriented-object extension of Bayesian networks called probabilistic relational models.

Probabilistic relational models have been first proposed by Friedman et al. (1999). They encompass multiple relations between class-object Bayesian networks, thus allowing a better representation between the different attributes (Torti et al., 2010). To allow such complexity they are defined on two levels (whereas Bayesian networks only requires one): the relational schema and

the relational model.

- **The Relational Schema** (Fig.1.5(a)) defines different groups of attributes called classes. Classes can be referenced to each other through so-called **reference slots**. The relations between classes are represented by oriented relations going from a class towards another, and are called **slot chains** (as they use reference slots to be defined). The orientation of these relations is important, as it will have an influence on the learning of the Relational Model: it is always unique (two classes cannot mutually refer to each other). At this point, only the group of attributes are defined and not the probabilistic dependencies.
- **The Relational Model** (Fig.1.5(b)) defines the probabilistic dependencies between the attributes. While the intra-classes relations are not constrained (in our example, relations between $\{X, Y\}$ or between $\{U, V, W\}$, meaning that inside these sets any probabilistic relation can be learned), the inter-classes' probabilistic dependencies are influenced by the slot chains: (1) if there is no slot chain between two classes, then their attributes cannot share probabilistic dependencies; (2) if there is one, then the direction of the probabilistic dependencies must follow the orientation of the slot chain.

**Ontological and Relational Schema's classes**

The term *class* is used both for the ontology and the relational schema indifferently. However, the distinction is important, especially in our work where they are used for different purposes. That is why, for the rest of this thesis, we distinguish them by annotating ontology's classes as $O$-classes.

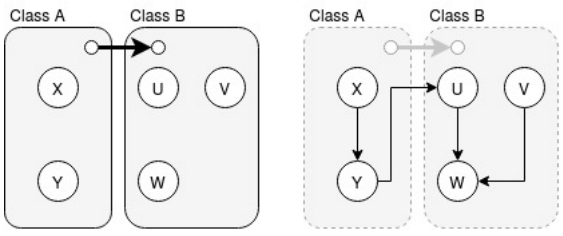



***Figure 1.5:*** **Structure of a Probabilistic Relational Model.** A Probabilistic Relational Model is described by two levels. (a) The relational schema defines groups of attributes as **classes**, and how those are related. (b) The relational model gives the probabilistic dependencies between the attributes.

Once both the relational schema and model are defined, the classes can be instantiated to build a probabilistic system. Using classes $A$ and $B$ defined in Fig.1.5, we can build a **system** as a combination of classes with respect to the reference constraints defined in the relational schema. Once defined, it can be instantiated as shown in the example presented in Fig.1.6. An instancia-

ted system is considered as a classical Bayesian network: therefore it shares the same properties presented in this section, and has an essential graph.

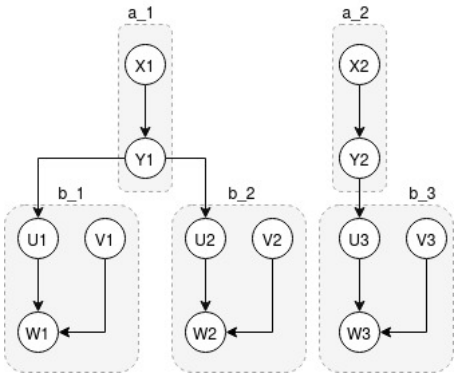



***Figure 1.6:*** **Example of a probabilistic relational model's system's instantiation.** Using the classes defined in Fig.1.5 we can instantiate this system, composed of two classes $A$ and three classes $B$.

Learning a probabilistic relational model can be an hard task, as one should both learn the relational schema and model. In this thesis, we focus on two parts:

- The translation of an ontology to the relational schema.
- The learning of the relational model from the relational schema and the data.

This last part is made easier by the fact that, in our work, the relational schema is not learned but built using an ontology. Since the relational schema is fixed, learning the relational model is similar to Bayesian network learning (Getoor and Taskar, 2007). Furthermore the relational schema also brings some information (shaped as constraints) that can improve the learning's results.

### 1.1.6 Learning under constraints

Learning a Bayesian network is an NP-hard problem whose difficulty drastically increases with the number of variables to consider. Learning blindly (i.e. without any external insight on the model) with an heuristic approach requires to test nearly all possible combinations, which is not optimal. On another hand, having some information about the model can help limit useless computation. We define such information as constraints we want to use to guide our learning towards a learned model closer to the reality.

Looking at Bayesian networks, numerous related works have established that using constraints with heuristic algorithms effectively improve structure (De Campos et al., 2009, Suzuki, 1996) and parameters (De Campos and Ji, 2008, Niculescu et al., 2006) learning. They may be more or less strict: in this thesis, the relational schema defines structural constraints as an ordering between the different variables.

**Definition 3:** *Precedence constraint*. Given two nodes $A$ and $B$, a precedence constraint from $A$ to $B$ implies that if $A$ and $B$ are connected by an arc, this has to be oriented such that $A \rightarrow B$.

We distinguish between **complete** and **partial** node ordering. We define a **complete node ordering**, if, for every two variables in the Bayesian network, there is a precedence constraint between them. On the contrary, we define a **partial node ordering** if the ordering is not complete.

**The peculiar case of Bayesian networks**

Given three nodes $A$, $B$ and $C$; the set $S$ of the two precedence constraints $A \rightarrow B$ and $B \rightarrow C$ is a partial node ordering since there is no precedence constraint between $A$ and $C$. However, a Bayesian network is a directed acyclic graph, meaning that cycles are forbidden: therefore we can easily infer from $S$ a third precedence constraint $A \rightarrow C$. As a consequence, in a Bayesian network, if for all nodes there is at least one precedence constraint such that all nodes can be placed in a chain such as $Node_1 \rightarrow Node_2 \rightarrow ... \rightarrow Node_N$, then the ordering is complete.

**Theorem 2:** *Bayesian networks' partial ordering*. If for all nodes of a BN there exists a precedence constraint such that the nodes can be placed in a chain, then the set of these precedence constraints is a complete ordering.

The K2 algorithm (Cooper and Herskovits, 1992) is an heuristic Bayesian network's structure learning algorithm that requires a complete ordering of its variables beforehand. This eases greatly the calculations since a lot of computations are left aside: if we have the precedence constraint $A \rightarrow B$ then we do not need to test $B \rightarrow A$. However, in order to apply K2 we need a thorough knowledge of the domain, which can be hard to come by, especially with numerous variables. Learning under partial knowledge has also been tackled, for instance, by Parviainen and Koivisto (2013) who propose a dynamic programming algorithm for learning Bayesian networks using partial precedence constraints to improve its efficiency.

In this thesis we focus on using a partial node ordering for guiding a heuristic algorithm to learn a Bayesian network.

### 1.1.7 Using Ontologies to learn Bayesian networks

Combining ontological knowledge in order to learn Bayesian networks has already been tackled by several works, since it offers a good compromise between asking for an expert input which

can be time-consuming and prone to mistakes (Druzdzel and Gaag, 2000), and relying entirely on the data which is not efficient as the algorithms are mostly score-based and do not take common sense into account. Masegosa and Moral (2013) propose, for instance, a method of edges selection in a Bayesian network using domain/expert knowledge. In this thesis, we will, however, focus on the combination of ontologies with Bayesian networks and probabilistic relational models. Most of the following works are based on similar methods, where the ontology brings knowledge in order to guide the structure building. Despite their great results, we wanted to focus, in our work, on a method able to transform any ontology; in order to do so, we had to consider pitfalls that felt prohibitive.

- **Dedication to specific ontologies.** A lot of works offer a transformation specifically designed for a single ontology, without possibilities of transfer towards another one. For instance, Bucci et al. (2011) use predefined templates to model support medical diagnosis, which cannot be extended to other ontologies; Helsper and Gaag (2002), Zheng et al. (2007) requires to construct a specific ontology to guide the construction of the Bayesian network structure.
- **Use of ontology's extensions.** The methods that require specific Bayesian ontologies' extensions such as the ones described in the introduction are limiting, since not all ontologies use them. This is the case for instance of the work presented by Devitt et al. (2006).
- **Direct translation of Object Properties.** Properties management can raise multiple issues. In a lot of the described approaches, the Bayesian network structure is not learned, but derived from the ontology: this is not what we aim to do as not all ontologies transcribe such direct dependencies. For instance, Fenz (2012) consider that the object properties directly serve as probabilistic dependencies if they are selected beforehand by an expert. Ben Ishak et al. (2011) take a similar stance, and assume that the ontology's properties are already causal in order to build an Object Oriented Bayesian Network.
- **Cardinality management.** To the best of our knowledge, no method addresses the case of object properties cardinality. Consider the classes *Teacher* and *Student*, and the object property *hasForTeacher*. By definition, a single teacher can have an unfixed number of students, which can be represented easily in the ontology, but not so simply in a probabilistic model. A further description of the problem and of one possible answer is given in chapter 6.

A few works have also been presented on the matter of combining object oriented Bayesian networks and ontologies (Ben Messaoud et al., 2011, Truong et al., 2005). Manfredotti et al. (2015) proposes the idea to map a probabilistic relational model's relational schema with an ontology to guide the learning of the relational model, which is the idea we develop in this thesis. However,

they do not explain how to do this translation.

## 1.2 Causality

As introduced in the introduction, our main motivation is the discovery of new knowledge in complex domains represented by an ontology. If the learning of a probabilistic model is a great help for the analysis of such domains, it is however not enough as it is only able to show probabilistic dependencies. In this section, we will present the main notions about causality and causal discovery.

### 1.2.1 Overview

Causality has been a topic of research for a long time (Reinchenbach, 1978, Suppes, 1970). In the early 1990s Pearl and others began to explore the meaning behind Bayesian networks edges (Pearl and Verma, 1991), which resulted in the works of Pearl (2009) that introduced **causal models**.

> **Definition 4:** *Causal models*. Directed acyclic graphs whose edges' orientations transcribe a causal dependency between the nodes.

Indeed, if Bayesian networks only show the correlation, causal models give causation, which is an efficient way to describe a model by offering a good insight on the relations between the variables and allowing to answer complex questions about them. In his recent book, Pearl and Mackenzie (2018) describe a causal ladder in order to distinguish the different levels of causal questions. It is composed of three rungs that go from the bottom to the top:

1. **Association.** The lowest rung is about **observing**, and answers the question: *"What if I see ...?"*. It is the most simple question, as it treats mostly correlation in the data. The authors place it at the level of machines and animals, as they deem them able to do that kind of computation.
2. **Intervention.** The middle rung is about **intervening**, and answers the question: *"What if I act on this factor?"*. Intervention is all about modifying a possible explaining factor to see if it changes the outcome. For instance, we know that umbrellas and rain are correlated; if we intervene on the umbrella factor (by preventing or forcing people to take one), will it have an influence on the weather? If so, then the umbrella is probably a causal factor for the rain; if not (as we should hope!), then it is not. This powerful reasoning tool tends however to be difficult to set, as not all events can be modified at will: the opposite experiment of the one we described in the example would be very hard to check, as it is nearly impossible to control the weather. Nevertheless, when it is possible, it can give a very astute overlook of

the causal knowledge encompassed in the model.

3. **Counter-factual.** The highest rung is about **imagining**, and answers the question: *"What would have happened, had I done that?"*. It is the most difficult question to answer, as it interrogates us on a set of events that did not happen, and on which we have no data. However, it can be answered using causal knowledge: I went outside with my umbrella and it was raining. Would it had rained, had I not taken my umbrella? If I use the causal knowledge discovered earlier (the umbrella has no causal effect on the rain), then I am able to answer: yes, it would have rained, umbrella or not. Answering counter-factual is the final goal for causal studies.

Classical Bayesian networks are at the first level. Our goal, in this thesis, is to constraint their learning with knowledge from the ontology and the expert in order to guide it with causal constraint towards a **causal Bayesian network** and thus climb the two last rungs.

> **Definition 5:** *Causal Bayesian Network*. Bayesian network whose graph is also a causal model. The edges transcribe both a probabilistic dependency and a causal effect, the head being the cause and the tail the effect.

As we have seen in Section 1.1.4, a same dataset can lead to learn several different but equivalent Bayesian networks. The goal of causal Bayesian networks learning is, then, to identify (if possible) among them the true model that respects every causal dependencies between the variables. This is not an easy task, that can be done using the second rung of the causality ladder, the interventions. However, this solution cannot be applied every time, for ethical, economical or practical reasons: we cannot for instance force a control group to smoke to assess whether it might have a causal impact on lungs cancer. This problem sparked the research field of causal discovery.

### 1.2.2 Causal discovery

Due to its implications in the field of explanations, causality is currently a trending topic in the computer science research community. However, it is far from an easy task. As any statistician comes to know, **"Correlation is not causation"**: a rainy weather and people having umbrellas usually come together, however we cannot assert that one event brings the other without external knowledge.

**Simpson's paradox**

First proposed by the statistician Edward Simpson in 1951, the Simpson's paradox is a blatant demonstration of the importance of external knowledge in the data and that correlation is not causation.

A study focuses on a group of sixty men and sixty women that took (or not) a drug, and whether they had an heart attack afterwards[a]. The data is presented in this table.

| | Control Group (No Drug) | | Treatment Group (Took Drug) | |
|---|---|---|---|---|
| | Heart attack | No heart attack | Heart attack | No heart attack |
| Female | 1 | 19 | 3 | 37 |
| Male | 12 | 28 | 8 | 12 |
| Total | 13 | 47 | 11 | 49 |

Simpson suspected that the drug might have a negative effect on male subjects. Looking at the general data, we can see that 22% of the control group suffered from an heart attack, while only 19% of the people who took the drug did: thus the drug seems to have a good impact on the heart condition. However, when looking at each individual on the base of their gender, we can asses that while 30% of men from the control group suffered from an heart attack, 40% did within the treatment group (a similar but of lesser magnitude conclusion can be drawn for women): this time, the drug seems indeed to be bad for the heart condition.

The paradox takes place in such a way it appears that **knowing the person's gender would render the drug harmful.** It demonstrates that depending the way it is processed, a same dataset can lead to opposite conclusions.

[a]This example is taken from *The Book of Why* (Pearl and Mackenzie, 2018), itself presenting the fictitious data set used by Simpson.

As a consequence, a data set in which we want to discover causal knowledge must answer some quality criteria about the **variables** and the **data quality**.

The variables are defined by the **causal sufficiency** criteria (Spirtes et al., 2000).

**Definition 6:** *Causal sufficiency*. Be $S$ a set of variables. We say that $S$ is causally sufficient if and only if for every common cause $P$ of two or more variables, $P$ either (1) is in $S$ itself or (2) has the same value for every variables in $S$.

The causal sufficiency insures to prevent a classic pitfall of causal discovery which is missing variables. For instance, a correlation can be found between one's shoe size and reading ability.

Using common sense, we know from experience that this is because both of these variables are explained by a third variable, the age: the older we are, the bigger our shoes are and the better we can read. Thus, attempting causal discovery with the truncated dataset would only lead to absurd discoveries. This is because when we consider only the shoe size and the reading ability, we are not in a causal sufficiency condition. The age is denoted as a **confounding factor**.

The quality of the data is sensitive to classical pitfalls such as missing data, selection bias, measurement error, non stationary or heterogeneous data and deterministic cases (Glymour et al., 2019). In the same way that a bad data processing can lead to erroneous conclusions, if not all possible events are present in the learning set, or if their proportion is altered and do not represent reality, then it is impossible to draw good causal discoveries.

As presented in Section 1.1.2, there are two Bayesian networks structure learning type of algorithms: heuristic and independence-based. In order to learn causal structures, this last category usually gives a better outlook on causality as they try to find the "true" orientation of the arcs. The principal algorithms are PC (Peter and Clark) and FCI (Fast Causal Inference), its variant that can handle confounding variables (Spirtes et al., 2000). MIIC (Multivariate Information based Inductive Causation) (Verny et al., 2017) is another algorithm that uses independence-based algorithms to obtain information considered as partially causal and thus allow to discover latent variables. These algorithm's drawback is that they do not easily allow constraints during their learning. There are approaches that combine these two families: GFCI (Greedy Fast Causal Inference) (Ogarrio et al., 2016) uses GES (Greedy Equivalent Search) to learn a structure, and FCI to prune it. However it does not offer a management of the different constraints as precise as we would like. That is why we preferred statistical methods and used essential graphs to deduce mandatory essential arcs.

Other works have already proposed the use of an essential graph to learn causal models: Hauser and Bühlmann (2014) propose two optimal strategies for suggesting interventions in order to learn causal models with score-based methods and the EG; Castelletti and Consonni (2020), Eberhardt (2008), Shanmugam et al. (2015) use an essential graph to build a causal Bayesian network while maintaining a limited number of intervention recommendations. These approaches do not require any external knowledge about the domain. In our case however, the data is encompassed in an ontology and is not sorted in a way such that a Bayesian network can be learned directly: as a consequence, while processing it, it can be interesting to see which information can be recovered from the ontology's semantic.

### 1.2.3 Ontologies and Explanation

The act of explaining usually echoes the causality. The Oxford dictionary defines an explanation as *"a statement, fact, or situation that tells you why something happened; a reason given for something"*.

The idea of using knowledge from ontologies in order to learn explanatory and causal models has been considered in several works. Ćutić and Gini (2014) presents the idea of integrating causal knowledge from ontologies for causal discoveries; Messaoud et al. (2009) offers a method to iterative causal discovery by integrating knowledge from beforehand designed ontologies to causal Bayesian networks learning. The main pitfall of these works is that they consider the properties within an ontology as causal, which is not the case each time and explains why we tried, in this work, to develop a more generic method.

One of the main interests of integrating knowledge from ontologies would be the ability to provide constructed and understandable explanations to why artificial intelligences have reach to this or that conclusion. Indeed, explainable artificial intelligences is a trending topic that rose with the multiplication of "black box" systems (i.e. systems that cannot explain their results in a manner humans can reason with). A lot of works have emerged on this topic (Ribeiro et al., 2016, Zhang and Chen, 2018), mostly about neural networks. On another hand, Bayesian networks, because of their graphical structure, are much easier to explain. Works can be done on their structure and sensitivity analysis (Lacave and Diez, 2002, 2004), but in general they are pretty straightforward to understand, since every node value can be explained with respect to the others. However, ontologies can bring another layer of explanation as they allow to introduce semantic notions to describe the relations between the nodes. Indeed, as we have seen before, there is no semantics behind the orientation of the probabilistic relations in a Bayesian network: $A \rightarrow B$ has the same validity (i.e. one model is not better than the other) as $B \rightarrow A$. However, if an ontology can give us an information such that $A \rightarrow B$ is not possible, then we would have more informations about the model and rule out all Bayesian networks with this relation.

As a consequence, ontologies can help to build arguments and explanations that models such as Bayesian networks would not be able to discover. For instance, Besnard et al. (2014) presents a tool combining ontological and causal knowledge in order to generate different argument and counterarguments in favor of different facts by defining enriched causal knowledge. They however did not cover the learning part.

## 1.3 Conclusion

Probability theory is the most efficient way to express events' likelihood. Bayesian networks, thanks to the addition of a graphical support, are well suited to describe complex models. However, if we want to couple their learning with ontological knowledge, they cannot express all the subtleties expressed in an ontology (e.g. $\mathcal{O}$-classes, instantiation, properties), especially in the case of ontologies where large relational informations are encompassed.

That is why, in this thesis, we chose to use probabilistic relational models: because of their two layers, the relational schema and the relational model, they allow a two-times learning which can better model our domains. They first define the classes and their attributes, then the probabilistic relations between them. Works have already been proposed on the combination of graphical models and ontologies. However, they were not generic enough to be adapted to any ontology, or were assuming criteria really hard to respect in real-life ontologies, such as the causality of the object properties. Once the model learned, we have proposed methods able to explain it even further with causal reasoning. Following the state of the art, we have decided to use the essential graph, that encompass all the Bayesian networks of the same Markov equivalence class. By introducing causal knowledge during the learning, we aim to give semantic and causal explanations of the domain, thanks to the knowledge base.

# CHAPTER 2

# LEARNING A PROBABILISTIC RELATIONAL MODEL FROM A SPECIFIC ONTOLOGY

**Contribution.**

Munch M., Wuillemin PH., Manfredotti C., Dibie J. and Dervaux S. Learning Probabilistic Relational Models using an Ontology of Transformation Processes. In: ODBASE 2017, Rhodes.

In this chapter, we present the first method we developed in order to couple ontologies and probabilistic models. For this work, we wanted to evaluate the gain that coupling ontologies with probabilistic relational models could bring when learning domain's model. That is why we first concentrated on a single specific ontology. To do so, we chose to study the Process and Observation Ontology $PO^2$ (Ibanescu et al., 2016). This ontology is dedicated to transformation processes, which are interesting to study as they can be complex (multiple variables, temporality).

**Section 2.1** first presents the transformation processes' specifities and which issues they raise for their modeling (2.1.1); in a second time, it presents in more details the ontology $PO^2$ and how it is efficient to represent transformation processes (2.1.2).

**Section 2.2** gives an overview of the algorithm we developed (2.2.1) and describes in more details its two key steps: the building of the relational schema (2.2.2) and the learning of the relational model (2.2.3).

**Section 2.3** presents our evaluation of this method, through a generation of synthetic datasets (2.3.1), the description of our protocol of evaluation (2.3.2) and comparison of the learning results (2.3.3).

**Section 2.4** discusses our algorithm and more specifically points two features from the studied domain that prevent our method from being generic: the concepts of explaining and explained attributes (2.4.1) and the temporality (2.4.2).

**Section 2.5** concludes this chapter.

## 2.1 Domain of application

In this chapter, we focus on a particular ontology that sparked our interest on combining probabilistic graphical models and ontologies: the $PO^2$ ontology, dedicated to transformation processes. We describe what we consider as a transformation process, before introducing the $PO^2$ ontology.

### 2.1.1 Transformation Processes

A transformation process is a generic way of describing a sequence of steps. By analogy, each step (i.e. operation) can itself be considered as a small thermodynamic system, with information entering (the inputs) and leaving (the outputs) it. The particularity -and difficulty- of analyzing them is that this flow of information is heterogeneous (different variables, scales) and time-dependent: each operation takes place within a specific time frame. This raises a substantial amount of variables that needs to be described, as illustrated in Fig.2.1.

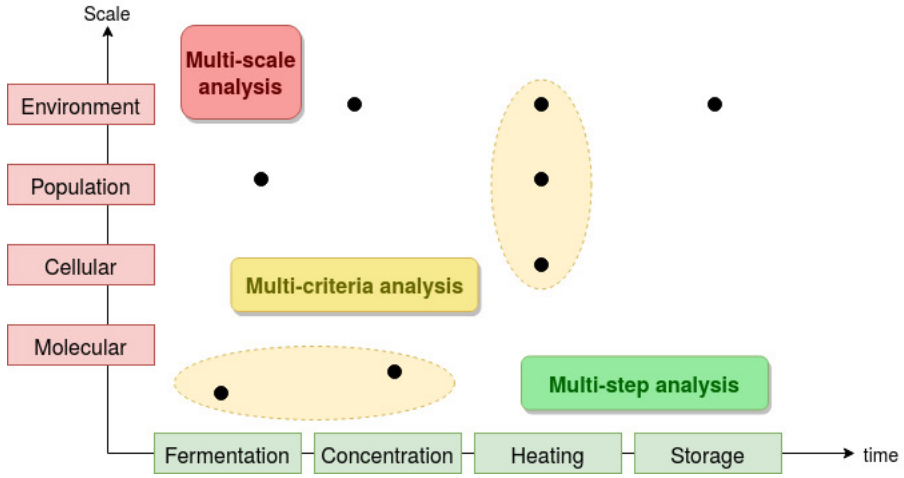



*Figure 2.1:* **Example of a transformation process in biology.** The multiple measures • represent a challenge to analyse, depending of the angle we wish to study them.

In this figure, an example of a transformation process is given. Each black dot • indicates a variable associated to the scale (ordinate axis) and the step (abscissa axis). In order to study this, it would be interesting to analyse how the variables influence each other: to do so, we have to conduct multiple analysis, across the steps, scale and variables themselves.

For the rest of this section, we define the following vocabulary when describing a transformation process:

- **A step** is an operation taking place at a specific time, that can be absolute (e.g. *"January the $1^{rst}$ 2020, 15:03 p.m."*) or relative (*"step 1 happens before step 2"*).
- **A process** is a succession of steps.
- **An attribute** is a variable that represents a measure taken during the transformation process.
- **An input** is an attribute defined at the beginning of the step that characterize it (e.g. an oven temperature).
- **An output** is an attribute that results from the step.

We will also consider that all transformation processes studied in this Section are well-defined: all inputs and outputs are useful to describe it. In most cases outputs are consequences of inputs. Two steps are considered identical if they share the same inputs and outputs, even if their value can vary. For instance, while baking a cake, we always define the step *Cooking* with *oven temperature* as an input. However, the temperature value itself can change, depending on what we intend to cook.

In conclusion, transformation processes are characterized by two kinds of complexity: time and scale. The goal of learning probabilistic model to represent a transformation process would be to be able to explain the different attributes with respect to the others in spite of these issues.

**Example 7.** Suppose an agro-food company wants to test a new process for baking their cakes. They have defined a process **Baking Cake**, composed of the following steps succession: **Mixing** → **Cooking** → **Tasting**. This process is detailed in Fig.2.2. Having a probabilistic modeling of it would allow us to verify the impact of the quantity of eggs, milk, oven's temperature on the taste of the final product.

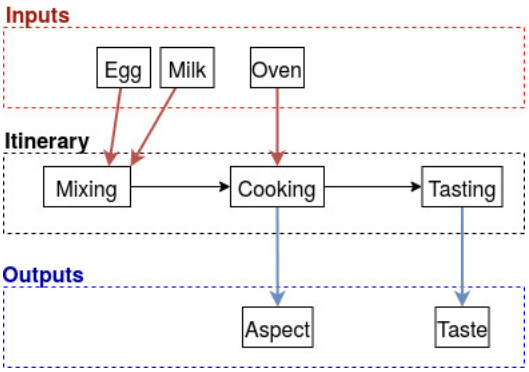



*Figure 2.2:* **Example of a transformation process.** The process is defined by three steps. We suppose that the cake dough is passed from one step to another, becoming an intrinsic input for **Cooking** and **Tasting**.

### 2.1.2 PROCESS AND OBSERVATION ONTOLOGY PO$^2$

The description of the transformation processes we gave in the previous section is coherent with the way the core ontology $PO^2$ is defined. Its conceptual components are mainly composed of four classes: the **Step**, **Participant**, **Observation** and **Attribute** classes. The participants and observations are described as follows:

- **Participants** represent the inputs. In the ontology, they can be of three natures: method (e.g. a measurement method, a selection method); mixture (e.g. a cake); and device (e.g. an oven). The mixture usually represents the product transformed throughout the whole process.
- **Observations** represent the outputs. They usually embody the different measures taken on the mixture and are characterized by multiple concepts: scale, sensory or computed observations...

The attribute is the intrinsic value associated to those participants and observations in order to describe them. They are themselves characterized by a numeric value and a unit, given by the datatype properties *po2:hasForValue* and *po2:hasForUnite*.

In this ontology, a process is a whole operation with a particular goal (e.g. baking a cake, transforming a product); if two processes share the same goal, they are the same. However, there are multiple means for a same end: a same goal doesn't mean that we need to always have the same succession of steps. As a consequence, the ontology differentiates the different succession of steps as **itineraries**. In an itinerary, each step is defined both by its participants and observations, and its relations to the other steps, for instance with the property *isBefore*. An overview of these conceptual components is given in Fig.2.3.

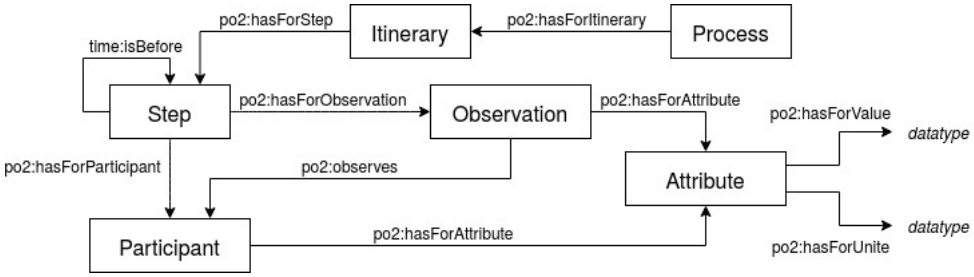



***Figure 2.3:* PO$^2$ main schema.** Selected view of the ontology with the classes that we have considered (Step, Observation, Participant, Attribute) and the datatype properties used.

However, in order to specify this ontology and adapt it to any transformation process, thus defining a **domain ontology**, one must first define new classes to introduce the particularities of the considered process. To do so, we use the *is-a* property who allows two classes to share the same specifications inside an ontology.

**Example 8.** In order to define the domain ontology of the Example 7, we need to define new steps, participants, observations and attributes. We define these new classes using the *is-a* property: **Mixing** *is-a* **Step**, **Egg** *is-a* **Participant**, ... As a consequence, Mixing is considered as a Step class and shares its specifications: it can be linked with *po2:isBefore* to other step classes, and can own participants and observations. The result is shown in Fig.2.4.

This defines a new domain ontology using the core ontology we presented in this section.

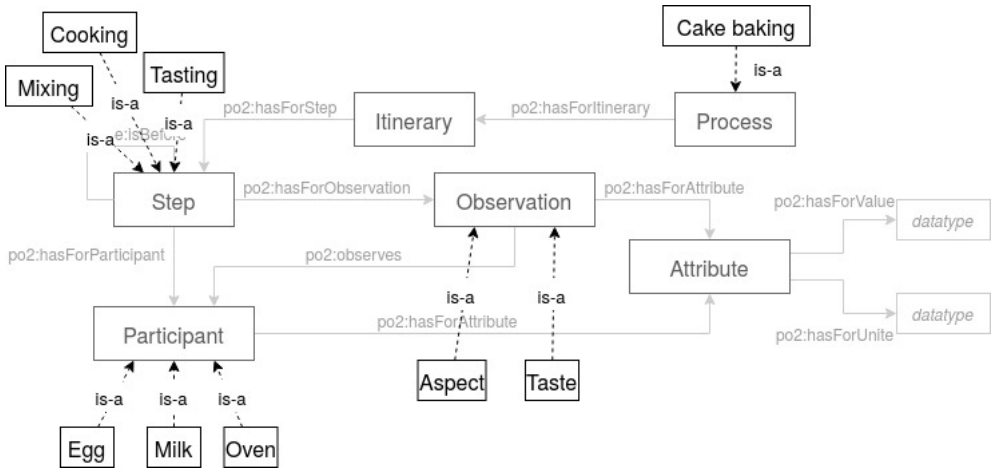



***Figure 2.4:* Example of a domain ontology using PO$^2$.** This is built using the transformation process example defined in Example 7. The new classes are specified thanks to the *is-a* property.

## 2.2 ON2PRM Algorithm

In this section we present our first contribution, the **ON2PRM algorithm** (**ON**tology **TO PRM**), whose purpose is to learn a probabilistic relational model from any domain ontology $\text{PO}^2_{domain}$ using the core ontology PO$^2$. Figure 2.5 gives an overview of its development.

### 2.2.1 Overview

The goal of learning a probabilistic relational model from a given domain is to be able to explain the different probabilistic dependencies (if they exist) between the variables. Yet, not all transformation processes can be transformed into a probabilistic relational model. Indeed, until now we have presented transformation processes as a succession of steps guided towards a same goal. However, probabilistic learning is based on a statistical counting: over all the presented evidences, the algorithm counts the repetition of each particular value to determine if they are frequent (high count) or not. This requires (1) that we have discrete data, (2) that the evidences themselves follow a same pattern and (3) that there are enough evidences so the statistical counting is meaningful.

For the first point, using PO$^2$ establishes that the only way to gather data is to use the Attribute class and the datatype property *po2:hasForValue*. In the following, we assume that all measures have been done using the same scale and that *po2:hasForUnit* is not necessary. The second point ensures that we compare what is comparable: as two processes with not the same goal cannot be compared, two different itineraries of a same process also cannot be compared, as they do not have the same steps, participants and observations. As a consequence, each itinerary of a same transformation process represents a different probabilistic relational model; and a probabilistic relational model can only be learned if there are multiple repetitions of this itinerary, which check the third point: the dataset used for the learning has to be complete and sound, with (if possible) no missing data. This last point is a strong pre-requisite, but is essential if we want our learning to make sense. Indeed, learning a probabilistic model requires to use a statistical counting of the different events, to evaluate their likelihood, whose measurement may be distorted by too many missing values.

Once this is established, we can use what we know of the transformation process domain in order to ease the learning. Indeed, we have to take into account two constraints:

**C1. Temporality.** The steps are recorded in time, which means that we can always define one in relation to the others. Since we aim to learn an explaining model, we take the assumption that a variable can always be influenced by one anterior or concomitant: **the past can explain the future, but not the contrary**. More specifically, this means that in our model, given three steps such that $Step_1 \rightarrow Step_2 \rightarrow Step_3$, attributes from $Step_2$ can be explained by attributes from $Step_1$ or $Step_2$ but not from $Step_3$.

**C2. Compartmentalization.** The conceptual difference between participants and observations is that participants are attributes fixed at the beginning of the step, while observations are attributes which are the result of this step. Therefore, the same way as past events can explain future events, participants can explain observations of the same step - but not the contrary.

This gives us a guideline for our new algorithm ON2PRM, that we will present in this section. The main idea behind this algorithm is to define a generic relational schema $\mathcal{RS}^{PO^2}$ that respects **C1** and **C2** that will be the same for all processes. Once defined, it can be used to automatically build the relational schema $\mathcal{RS}^{PO^2}_{domain}$ for any domain ontology of PO$^2_{domain}$. The relational model $\mathcal{RM}_{domain}$ can then be learned, using the constraints given by $\mathcal{RS}^{PO^2}_{domain}$.

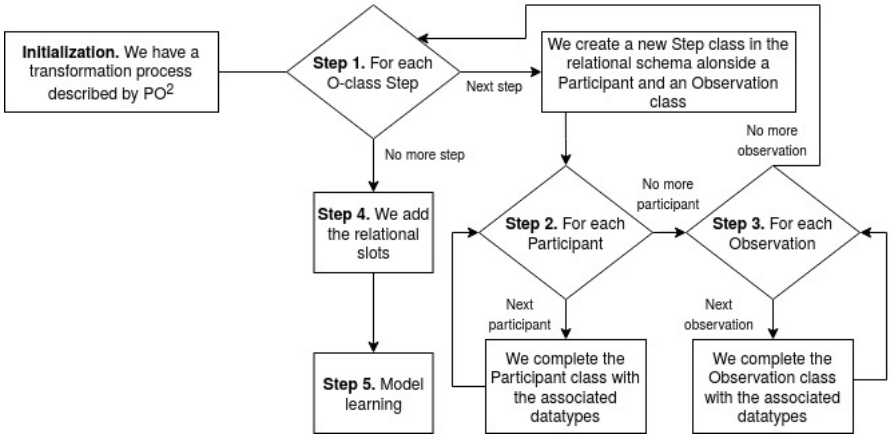



*Figure 2.5:* **Overview of the ON2PRM Algorithm.** Overview of the algorithm allowing to learn a probabilistic relational model from any domain ontology $\text{PO}^2_{domain}$ using the core ontology $\text{PO}^2$.

### 2.2.2 Building the Relational Schema

Given the constraints **C1** and **C2**, we can define the generic relational schema presented in Figure 2.6. In this figure, we describe two generic steps happening at times $t$ and $t+1$, both with participants and observations.

**C1** ensures that events can have a probabilistic relation towards concomitant or future ones, but not on past events. This constraint is translated into a **temporal** relational chain between the Step classes, with an orientation from $Step_t$ to $Step_{t+1}$.

**C2** ensures that participants can have a probabilistic relation towards observations of the same step, or participants and observations of the next steps, but not on past observations or past participants. This constraint is translated both by the **explanatory** relational chain between participant and observation, and also by the temporal one between $Step_t$ and $Step_{t+1}$.

When building a specific relational schema $\mathcal{RS}^{PO^2}_{domain}$ for a given domain, we define as many step classes as there are steps in the domain ontology. Each is linked with temporal relational chains that we determine using the *po2:isBefore* object property.

**Example 9.** In the cooking transformation process of Example 7, we have three steps, therefore three classes in $\mathcal{RS}^{PO^2}_{cooking}$. They are linked such that for instance **Cooking** has a relational chain towards **Tasting** and **Mixing** has one towards **Cooking**. However, using the **Cooking**' reference slot allows **Mixing** an other relational chain towards **Tasting**. The resulting relational schema is presented in Figure 2.7.

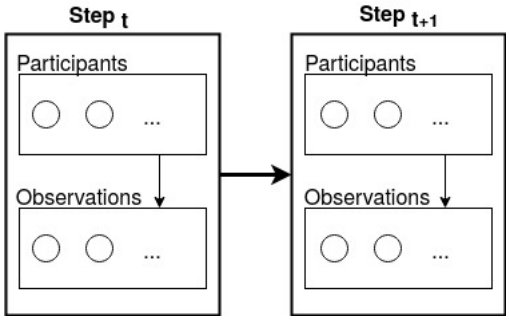



***Figure 2.6:*** **Generic relational schema $\mathcal{RS}^{PO^2}$.** This generic relational schema is designed so it respects transformation processes constraints. All participants and observations of a same step are linked by an explanatory relational chain; all steps are linked together by a temporal relational chain.

**Markov Property**

An event follows the Markov property if and only if its future state is not influenced by its past state. In this case, the future only depends on the event's current state. More formally, we note that

$\forall t \in [0; N], \forall k \in [0; t]$, P($X_{t+1} \mid X_t$,$X_{t-1}$,... $X_{t-k}$) = P($X_{t+1} \mid X_t$)

It can be generalized for a transformation process: in this case, to determine the state of a step, we only need to know the state of the one before. By default, thanks to the temporal slot chains between the different steps, each step has "access" to more than the previous step, and thus in theory a variable in one step can have a conditional probability depending on the variables from any step before. It would be easy to restrict the learning order so to respect the Markov property: however, it is a particular case that we do not consider in our works. As a consequence and unless mentioned otherwise, we **do not restrict the slot chains during the learning of the probabilistic relations**: if a relational schema's class $A$ has access through a slot chain to a class $B$, then the attributes of $A$ can have a probabilistic relation with the attributes of class $B$.

While defining the steps, we look at its participants, observations, and their associated attributes. For each, we define an attribute in $\mathcal{RS}^{PO^2}_{domain}$ such that:

- The name of the attribute is composed as follows: *name of the participant/observation + name of the attribute*,
- The value of the attribute is the value given by the datatype property *po2:hasForValue*,
- The attribute is associated to the subclasses Participant or Observations depending if it is linked to a Participant or an Observation through the object property *po2:hasForAttribute*.

**Example 10.** We look at the **Cooking** Step, which has for participant *oven* and for observation *aspect*. We suppose the oven has for attribute *Temperature* and the aspect has the attribute *Color*. As a consequence we define two variables: *Temperature of the Oven*, that goes in the subclass Participant of the Cooking class and *Aspect color*, that goes in the subclass Observation.

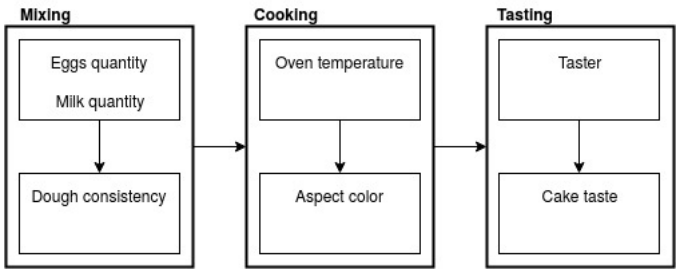



***Figure 2.7:*** **Example of a relational schema built from the domain ontology example.** This is built using the transformation process example defined in Example 7. It is composed of the three steps we defined and the different attributes that describe the participants and observations.

The role of $\mathcal{RS}^{PO^2}_{domain}$ is to guide the direction of the (possible) learned relations, in order to constraint them towards a probabilistic model that respects the constraints **C1** and **C2** given by the transformation processes and PO$^2$.

### 2.2.3 Learning the Relational Model

Once $\mathcal{RS}^{PO^2}_{domain}$ built, we can learn the relational model $\mathcal{RM}_{domain}$. As described in Chapter 1, the learning of a probabilistic relational model while knowing the relational schema is comparable to learning a Bayesian network under constraints. The interest of the relational schema is that it defines a partial node ordering for the attributes. As a consequence, each class defined in $\mathcal{RS}^{PO^2}_{domain}$ can be learned as a small Bayesian network, while respecting the following learning order $1 \rightarrow 2 \rightarrow 3$:

1. **The previous attributes.** All attributes (participants and observations) attached to the previous steps (if they exist) are grouped in a same set.
2. **The current participants.** All attributes attached to the considered step's participants are grouped together.
3. **The current observations.** All attributes attached to the considered step's observations are grouped together.

The probabilistic relations learned are then kept in $\mathcal{RM}_{domain}$. However, it is important to note that we only keep the ones that are engaged with variables (participant or observation) of the step that we are learning: all relations learned between the **previous attributes** set are not kept: this is due to the fact that these attributes are all mixed in a same group without compartmentalization

between them.

**Parallel steps**

Until now we have only presented small transformation processes composed of a linear succession of steps. However, PO$^2$ allows the creation of parallel steps, meaning steps that are defined before a same steps, but that are not related. This is usually the case for transformation processes that require multiple preparations: for instance, a mixture prepared in a step **Mixing** can require two different ingredients respectfully prepared in steps **Ingredient 1 Preparation** and **Ingredient 2 Preparation**. Those two do not interact, so they do not share a *po2:isBefore* object property; and during the specific relational schema construction, they are not linked through a relational chain, although they both have a relational slot with **Mixing**.

This is why, when learning the **Mixing** class for the relational model, we do not consider possible learned probabilistic relations between attributes of the set of the previous attributes (in our example, **Ingredient 1 Preparation** and **Ingredient 2 Preparation**). Even if they both are before the same step, they do not interact, and cannot share a probabilistic relation.

Once each class is learned in the relational model, the classes of the probabilistic model are known and defined, it becomes possible to instantiate the system and build a Bayesian network. In the rest of this chapter, we are going to focus on the evaluation of this algorithm and on how it helps to improve the learning and its result compared to the classical naive learning of a Bayesian network.

## 2.3 Evaluation

In order to evaluate our algorithm, we have defined the following protocol that consists of four parts:

1. **Random creation of transformation processes.** We generate a succession of steps, with different participants and observations whose attributes share probabilistic relations. In real life, this constitute the **ground truth** we aim to approach and model.
2. **Generation of synthetic data sets**. They are generated following the probability distributions described in the transformation processes and represent the experimental data. They are presented in two fashions: a raw one which is a simple data table and a structured one represented in a dedicated domain ontology using PO$^2$.
3. **Learning a Bayesian network.** Since (1) an instantiated probabilistic relational model is equivalent to a Bayesian network, and (2) it is harder to learn a probabilistic relational model

than a Bayesian network, we have chosen to compare our algorithm to Bayesian network learning methods. We define arbitrarily two methods $\mathcal{M}_1$ and $\mathcal{M}_2$ that both correspond to the combination of classical scores and an heuristic algorithm: $\mathcal{M}_1$ represents *Greedy Hill Climbing* (Gámez et al., 2011) with the BIC score (Schwarz, 1978), and $\mathcal{M}_2$ the *Local Search with tabu list* (Holland, 1975) with the BDeu score (Buntine, 1991). For each of these methods, we test two kinds of learning:

- **Naive learning.** We simply use the raw data set generated in part 2, and learn a Bayesian network without constraining or bringing knowledge from the fact that we are dealing with a transformation process. We denote it $\mathcal{M}_1$ or $\mathcal{M}_2$, depending on the method used.
- **Learning with ON2PRM.** We use the algorithm as defined in Section 3.2 on the structured data generated in part 2, with either $\mathcal{M}_1$ or $\mathcal{M}_2$ while learning the relational model. We denote ON2PRM($\mathcal{M}$) to indicate that we have used the learning method $\mathcal{M}$.

4. **Evaluating and comparing the scores of the resulting Bayesian networks.** We define scores to assess the proximity of the results with the ground truth.

The next subsections will give more details about this protocol and will present the results.

### 2.3.1 Generation of synthetic data sets

We generate random transformation processes based on the definitions given in Section **??**: a transformation process is a succession of different steps, each step being composed of inputs and outputs. We consider for the following the vocabulary of the PO$^2$ ontology since its conception of transformation processes is similar to our definition.

Since one of our motivation for studying transformation processes was their complexity, we had to take into account specific parameters to evaluate their diversity. Indeed, limiting the study to only one or two transformation processes without characterization would have been harmful for the results, considering one cannot represent alone this diversity spectrum. Thus, we define five **process complexity criteria** to qualify a process:

1. the number $s$ of steps in a process;
2. the maximal number $p$ of parallel steps, representing how many direct parents a step can have;
3. the number $n$ of attributes in a class;
4. the number $m$ of modalities for the attributes, meaning the number of value an attribute can have;

5. the maximum number $d$ of probabilistic dependencies an attribute may have.

The higher the process complexity criteria are, the harder to learn the corresponding probabilistic models are. As a matter of fact, during learning of a the relational model:

- a high number of steps $s$ induces more probabilistic relational model's classes to learn;
- a high number of parallel steps $p$, attributes $n$ and probabilistic dependencies $d$ induces more possible links to evaluate;
- a high number of modalities $m$ induces a more difficult learning, due to bigger conditional probability tables to compute.

These difficulties are also retranscribed during the Bayesian networks learning, where more steps increase the number of attributes and possible probabilistic relations between them. In the following, we make the hypothesis that the process complexity criteria are better addressed by ON2PRM where the ontology semantic knowledge reduces the learning's complexity. Therefore, we argue that if the results of our approach outperforms that of a standard method for simple processes, it will also have better results in learning more complex processes. Considering this assumption and to be as close as possible to the modeling of real transformation processes, we decided to fix two process complexity criteria: $m = 2$ (i.e. binary attributes) and $d = 3$, and to have three criteria that vary: $s \in \{3, 5, 8\}$, $p \in \{1, 2, 3\}$ and $n \in \{2, 4\}$. This leads us to 16 possible configurations, since the case $\{s = 3 \cap p = 3\}$ (i.e. a process composed of three parallel steps without interaction) is not relevant to study.

Even with all these parameters fixed, the number of possible transformation process is high: Figure 2.8 illustrates this notion by giving an example of randomly decided inter-steps (a) and inter-attributes (b) relations. Finally, even with a same structure, the probabilistic dependencies between the attributes were randomly defined. In the end, we generated 10 transformation processes for each configuration, meaning a total of 160 processes.

While studying transformation processes, we in reality never have access to these models: as a consequence, in order to mimic the reality of experiments we have to draw a dataset from random sampling, where each sample represents a complete experiment. These datasets are presented in two forms:

- **A raw form**, where the experiments are just presented as a classical data set (variables and values for each sample), without any compartmentalization between the attributes.
- **A structured form**, where every experiment is recorded in a knowledge base built following the generation of the associated transformation process from which the experiments are extracted. These domain ontologies are also characterized by the same process complexity degree criteria defined above: the number of steps is fixed by $s$, the number of participants


(a) Fixed $s = 5$ and fixed $p = 3$ (b) Fixed $n = 2$ and fixed $d = 3$

***Figure 2.8:*** **Variety of the transformation processes structures.** (a) shows the diversity of inter-classes relations: even with a same skeleton, the steps can be linked differently. On another hand, (b) shows that even with the steps and relations fixed, the probabilistic dependencies between the attributes can also be different.

and observation attributes by $n$.

For each of the 160 built transformation processes, we generate 100 datasets of four different sizes: 50, 100, 150 and 200, which gives us 64,000 datasets. The size of those datasets have been voluntarily kept low, as usually in real life experiments are hard to realize. In this experiment, we have chosen to verify how well our algorithm would perform with small datasets, as good learning are usually harder to perform under those conditions. The overall plan of experiments is shown in Figure 2.9.

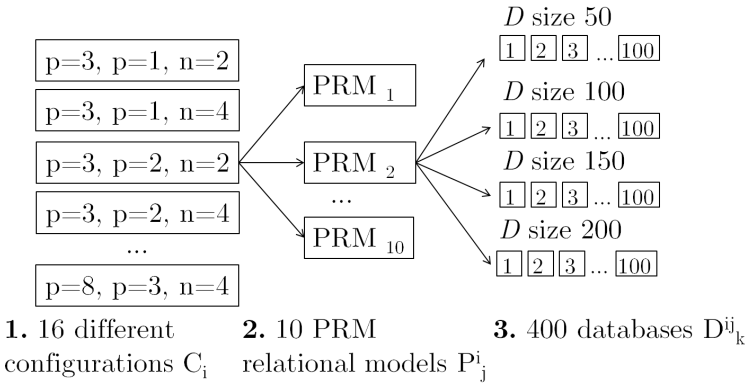



***Figure 2.9:*** **Plan of experiments.** For each configuration of process complexity degree criteria (1), we generate 10 transformation processes (2). For each of these, we then generate 100 datasets of different sizes (3) under two forms (raw and structured).

### 2.3.2 EXPERIMENTS

As mentioned before, the generated transformation processes represent the goal, the ground truth we wish to approach. Thus, the closer the models we learn (using the generated datasets) are to them, the better these models represent the reality, and as a consequence the better is the learning

*Table 2.1:* **Heuristic used to compare two BNs.** TN: True negative. FN: False negative. TP: True positive. FP: False positive

| Learned \ Model | $\emptyset$ | $\rightarrow$ | $\leftarrow$ |
|---|---|---|---|
| $\emptyset$ | TN | FN | FN |
| $\rightarrow$ | FP | TP | FN |
| $\leftarrow$ | FP | FN | TP |

method. Thus, we define an estimation criteria to evaluate the proximity between two Bayesian networks. This can be done rather by studying either the **probability distributions** (using for instance the Kullback–Leibler divergence (Kullback, 1959)), or the **structural differences** (with the recall, precision and f-score measures (Kent et al., 1955)). The first allows us to compare between the models the probability distributions of a same variable and see how it was impacted by the different learning methods (i.e. if its probability is still the same or not despite the possible different structures). On the contrary, the second study focuses only on the structure and not on the probability distribution: it compares whether an arc is missing or not, its orientation, etc.

Since our extended goal goes beyond the simple fact of finding probabilistic dependencies, we also wish to have a true semantic modeling of the domain able to explain how the different attributes interact with each other. In this focus, the structural analysis brings us more information, since it also allows an understandable (from the domain's knowledge viewpoint) interpretation of the relations. Indeed, due to the semantic value added, edges orientation is crucial: that is why we consider the presence of arcs as well as their orientation while evaluating the performance.

As a consequence, our evaluation protocol is based on the structural differences between the ground truth and the learned model using ON2PRM($\mathcal{M}$) or directly $\mathcal{M}$. In order to compute these differences, we have to count the number of true positive $TP$ and true negative $TN$ (i.e. right learning), and false positive $FP$ and false negative $FN$ (i.e. wrong learning). These are defined following the heuristic reported in Table 2.1.

This helps us to compute the following scores:

- **Recall** $\mathcal{R}$ estimates the number of links found out of the total we have to find: $\mathcal{R} = \dfrac{TP}{TP+FN}$
- **Precision** $\mathcal{P}$ estimates the proportion of true links among the ones found: $\mathcal{P} = \dfrac{TP}{TP+FP}$
- **F-score** $\mathcal{F}$ computes the mean value of recall and precision: $\mathcal{F} = \dfrac{2\mathcal{R}\mathcal{P}}{\mathcal{R}+\mathcal{P}}$

For each dataset, we realize two learnings with ON2PRM($\mathcal{M}$) and two learnings with $\mathcal{M}$. Then, we compute the F-score for each, compared with the original transformation process. For each of the 160 processes, we then have 400 F-scores, 100 for each data set size. In order to have a comprehensible and synthetic analysis, we present, in the next section, the computed average of the F-score for each of the 16 categories of transformation processes and the four sizes of dataset.

*Table 2.2:* **Variation of the mean F-score in function of different parameters tested with a dataset of size 50 with 100 repetitions.** The value between brackets represents the confidence interval at 99%. **bold**: highest value in column, *italic*: lowest value in column. $s$: number of steps, $p$: maximal number of parallel steps, $n$: number of attributes

| $s$ | $p$ | $n$ | ON2PRM($\mathcal{M}_1$) | $\mathcal{M}_1$ | ON2PRM($\mathcal{M}_2$) | $\mathcal{M}_2$ |
|---|---|---|---|---|---|---|
| 3 | 1 | 2 | **0.40** [0.03] | 0.27 [0.03] | 0.56 [0.03] | **0.33** [0.03] |
| | | 4 | 0.33 [0.02] | 0.25 [0.02] | 0.45 [0.02] | 0.26 [0.02] |
| | 2 | 2 | *0.24* [0.03] | *0.17* [0.03] | 0.43 [0.03] | 0.26 [0.03] |
| | | 4 | 0.25 [0.01] | 0.20 [0.01] | *0.38* [0.02] | 0.24 [0.02] |
| 5 | 1 | 2 | **0.40** [0.02] | **0.29** [0.02] | 0.54 [0.02] | 0.27 [0.02] |
| | | 4 | 0.30 [0.01] | 0.22 [0.01] | 0.43 [0.01] | 0.22 [0.01] |
| | 2 | 2 | 0.37 [0.02] | 0.27 [0.02] | 0.54 [0.02] | 0.27 [0.02] |
| | | 4 | 0.29 [0.01] | 0.21 [0.01] | 0.42 [0.01] | 0.21 [0.01] |
| | 3 | 2 | 0.37 [0.02] | 0.28 [0.02] | 0.52 [0.02] | 0.27 [0.02] |
| | | 4 | 0.28 [0.01] | 0.22 [0.01] | 0.41 [0.01] | 0.21 [0.01] |
| 8 | 1 | 2 | 0.45 [0.01] | **0.29** [0.02] | **0.58** [0.01] | 0.25 [0.01] |
| | | 4 | 0.31 [0.01] | 0.21 [0.01] | 0.43 [0.01] | *0.17* [0.01] |
| | 2 | 2 | 0.37 [0.02] | 0.25 [0.02] | 0.52 [0.02] | 0.22 [0.01] |
| | | 4 | 0.31 [0.01] | 0.22 [0.01] | 0.44 [0.01] | 0.18 [0.01] |
| | 3 | 2 | 0.34 [0.02] | 0.24 [0.02] | 0.52 [0.02] | 0.22 [0.01] |
| | | 4 | 0.31 [0.01] | 0.22 [0.01] | 0.43 [0.01] | 0.18 [0.01] |

### 2.3.3 RESULTS

#### Comparison on datasets of size 50

Table 2.2 presents the results for the two methods $\mathcal{M}_1$ and $\mathcal{M}_2$, with and without ON2PRM, on the learning with the smallest datasets (size 50). In every case, the results of learning with ON2PRM are significantly better, with a confidence interval at 99%. This goes into the sense of our hypothesis: keeping the data compartmentalized using the ontology's semantic enhances the learning. This can be explained by the fact that the relational schema respects $C1$ and $C2$: while using it, the algorithm already has some direction constraints (temporal and structural), and does not test useless hypothesis. It drastically reduces the number of possibilities the method $\mathcal{M}$ has to consider. As a consequence, even with a very reduced dataset size, we obtain a far better result.

#### Impact of the data set size

Table 2.3 presents the evolution of the F-score depending on the dataset size for two different combination of transformation processes complexity criteria, with (a) a low-complexity and (b) a high-complexity. Once again, we can assess that ON2PRM($\mathcal{M}$) is better than $\mathcal{M}$. We can also observe that the F-score increases with the dataset size, which is coherent: the more examples we have, the better and more precise the learning is. Moreover, we can also assess that the more complex (process complexity criteria wise) a transformation process is, the lower the F-score is, which is also coherent: since there are more parameters to test (more steps, more possible connections),

then the room for error is more important.

### Differences between the methods

As shown in Table 2.3, recall and precision are both as significant as F-score; however depending on the methods, performance varies. Precision tends to be, in fact, better with $\mathcal{M}_1$, while recall is better with $\mathcal{M}_2$. Since the difference between recall and precision for $\mathcal{M}_2$ is smaller than for $\mathcal{M}_1$, it explains why $\mathcal{M}_2$ has the best F-score.

***Table 2.3:*** **Comparison of performances for recall, precision and F-score for $\mathcal{M}_1$ and $\mathcal{M}_2$ with different sizes of the dataset.** The value between brackets represents the confidence interval at 99%.

| Method | Length | Recall | | Precision | | Fscore | |
|---|---|---|---|---|---|---|---|
| | | ON2PRM($\mathcal{M}$) | $\mathcal{M}$ | ON2PRM($\mathcal{M}$) | $\mathcal{M}$ | ON2PRM($\mathcal{M}$) | $\mathcal{M}$ |
| M1 | 50 | **0.26** [0.04] | 0.16 [0.03] | **0.95** [0.05] | 0.81 [0.09] | **0.4** [0.05] | 0.27 [0.04] |
| | 100 | **0.39** [0.04] | 0.24 [0.04] | **0.97** [0.02] | 0.87 [0.07] | **0.54** [0.05] | 0.37 [0.05] |
| | 150 | **0.47** [0.04] | 0.28 [0.04] | **0.97** [0.02] | 0.86 [0.06] | **0.62** [0.04] | 0.41 [0.05] |
| | 200 | **0.51** [0.04] | 0.31 [0.04] | **0.97** [0.02] | 0.88 [0.06] | **0.66** [0.04] | 0.44 [0.05] |
| M2 | 50 | **0.44** [0.04] | 0.27 [0.04] | **0.82** [0.04] | 0.46 [0.05] | **0.56** [0.04] | 0.33 [0.04] |
| | 100 | **0.53** [0.04] | 0.33 [0.04] | **0.90** [0.03] | 0.61 [0.06] | **0.66** [0.04] | 0.42 [0.05] |
| | 150 | **0.57** [0.04] | 0.38 [0.05] | **0.92** [0.03] | 0.69 [0.05] | **0.70** [0.03] | 0.48 [0.05] |
| | 200 | **0.61** [0.04] | 0.4 [0.04] | **0.94** [0.02] | 0.72 [0.05] | **0.73** [0.03] | 0.50 [0.05] |

(a) Parameters of the process: $s$ = 3, $p$ = 1, $n$ = 2

| Method | Length | Recall | | Precision | | Fscore | |
|---|---|---|---|---|---|---|---|
| | | ON2PRM($\mathcal{M}$) | $\mathcal{M}$ | ON2PRM($\mathcal{M}$) | $\mathcal{M}$ | ON2PRM($\mathcal{M}$) | $\mathcal{M}$ |
| M1 | 50 | **0.19** [0.01] | 0.13 [0.01] | **0.91** [0.02] | 0.61 [0.03] | **0.31** [0.02] | 0.22 [0.01] |
| | 100 | **0.29** [0.01] | 0.21 [0.01] | **0.93** [0.01] | 0.73 [0.03] | **0.44** [0.01] | 0.33 [0.02] |
| | 150 | **0.36** [0.01] | 0.27 [0.01] | **0.94** [0.01] | 0.77 [0.02] | **0.52** [0.02] | 0.40 [0.02] |
| | 200 | **0.42** [0.01] | 0.32 [0.02] | **0.94** [0.01] | 0.8 [0.02] | **0.58** [0.02] | 0.46 [0.02] |
| M2 | 50 | **0.33** [0.02] | 0.19 [0.01] | **0.61** [0.02] | 0.16 [0.01] | **0.43** [0.02] | 0.18 [0.01] |
| | 100 | **0.42** [0.02] | 0.26 [0.02] | **0.78** [0.02] | 0.32 [0.02] | **0.54** [0.02] | 0.29 [0.02] |
| | 150 | **0.48** [0.02] | 0.32 [0.02] | **0.84** [0.02] | 0.44 [0.02] | **0.61** [0.02] | 0.37 [0.02] |
| | 200 | **0.52** [0.02] | 0.36 [0.02] | **0.87** [0.01] | 0.52 [0.02] | **0.65** [0.01] | 0.42 [0.02] |

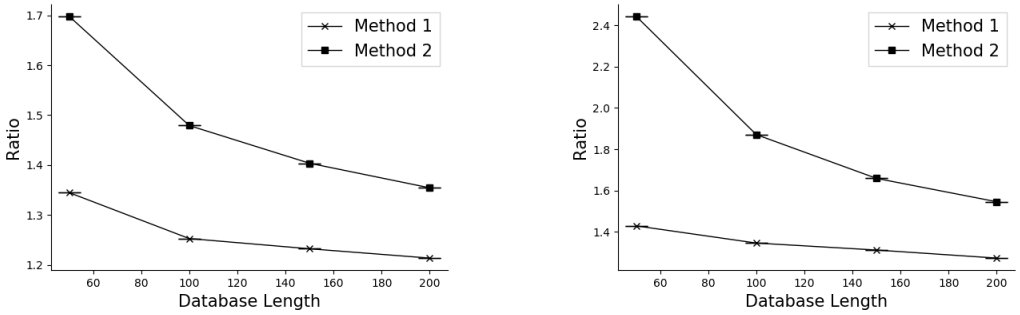
(b) Parameters of the process: $s$ = 8, $p$ = 3, $n$ = 4

### Evolution with dataset size

Even with few data a difference between the two learning approaches appears. Moreover while raising the size of the dataset, every score increases. In order to quantify and compare the performance of learning with ontology and without, we introduce the following ratio $\mathcal{R}_T$ of the performances: $\mathcal{R}_T = \frac{\text{performance with ON2PRM}(\mathcal{M})}{\text{performance with } \mathcal{M}}$

The more $\mathcal{R}_T$ is above 1, the more the learning with the ON2PRM algorithm is efficient. We have used this value to compare the evolution of scores with processes complexity and the different complexity criteria defined (number of step $s$, number of parent $p$ and number of attribute $n$). Figure 2.10 illustrates the evolution of the ratio $\mathcal{R}_T$ for two processes, (a) a simple and (b) a complex. As we can see, it is always above 1, confirming that the learning with ON2PRM($\mathcal{M}$) is more efficient. However, we can also asses its decrease, which explains that the more data we have, the lesser the difference between ON2PRM($\mathcal{M}$) and $\mathcal{M}$ is, meaning that the additional

data complements $\mathcal{M}$ for its lack of structuration. Depending on the methods this drop can be narrower or wider: while $\mathcal{M}_1$ stays practically stable $\mathcal{M}_2$ drops faster. Moreover the ratio varies equally with the complexity for $\mathcal{M}_2$: ON2PRM efficiency is higher with a complex process, which translates into a higher $\mathcal{R}_T$.




(a) Simple process ($s$ = 3, $p$ = 1, $n$ = 5) (b) Complex process ($s$ = 8, $p$ = 3, $n$ = 5)

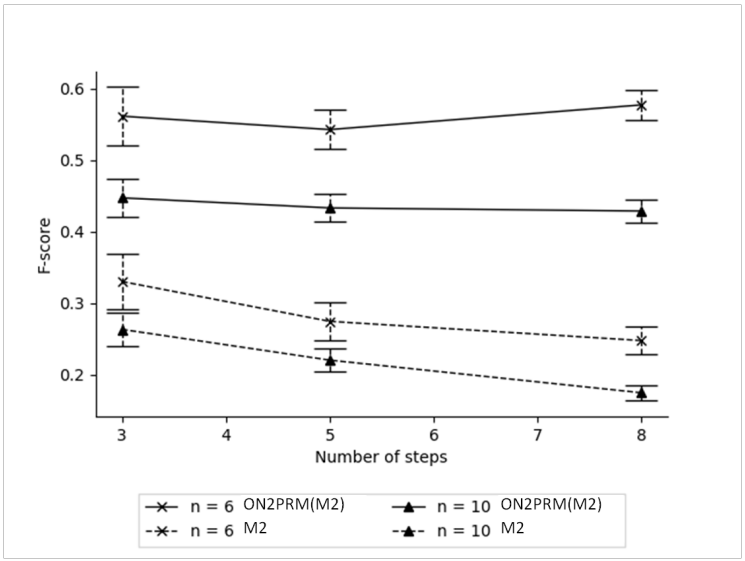
***Figure 2.10:*** **Evolution of F-score ratio for two different processes with the dataset length.**

#### Influence of the numbers of steps and attributes

Finally, we want to analyse the impact of the number of steps and attributes, which are the criteria the most impacted by the compartmentalization of the PRM. Figure 2.11 presents the evolution of the F-score in function of the number of steps, for two different numbers of attributes given. We can assess that (1) the more attributes there are, the lower the F-score is; and (2) given the confidence interval, it does not appear that the number of steps has a significant influence on the F-score, even if the global tendency seems to be a stagnation for ON2PRM($\mathcal{M}$) and a decrease for $\mathcal{M}$. As shown in the figure, if computed, the ratio $\mathcal{R}_T$ appears to augment with the number of steps, which shows that the efficiency of ON2PRM($\mathcal{M}$) also increases. This is in accordance with what has already been shown: since the relational schema helps to have a better compartimentalization, a higher number of steps will be computed the same way for ON2PRM($\mathcal{M}$), whereas it will add more complexity for $\mathcal{M}$ alone that cannot differentiate between the different steps.

Following the results presented in this section, we can conclude that our algorithm is efficient when learning a probabilistic model of a domain represented in the PO$^2$ ontology: the graphs structures are indeed closer to the ground truth than when learning with a naive learning.

However, the ON2PRM algorithm has been built for domain ontologies using the core ontology PO$^2$, which is limited. Indeed, this algorithm asks for some prerequisites that are not always guaranteed in other common ontologies. Next section will discuss its limits and how it could be expanded to other applications.




| s | 3 | | 5 | | 8 | |
|---|---|---|---|---|---|---|
| n | 3 | 5 | 3 | 5 | 3 | 5 |
| Ratio | 1.70 | | 1.95 | | 2.35 | 2.45 |

***Figure 2.11:*** **Evolution of F-score in function of n (p = 1, dataset size = 50) and ratio evolution in function of n and s for $\mathcal{M}_2$.**

## 2.4 Discussion

Since the only difference between learning with ON2PRM and learning without stems from the relational schema, we can safely assess that the compartmentalization brought by this schema helps improve the learning results, which seems logical: the more information we know about the domain, the closer to it our learning will be. Thus the relational schema seems to be the perfect way to introduce all ontology's knowledge when coupling ontologies with probabilistic relational models.

However, the one we introduced in this chapter is not general: it is specific to the PO$^2$ ontology, since the classes defined in $\mathcal{RS}^{PO^2}$ are directly recovered from its concepts: **Step**, **Participant**, **Observation** and **Attribute**. Moreover, they are also not direct translations, since the Attribute class is translated into an attribute and not a class in the relational schema. All of these choices have been made consciously and illustrate the good grasp of the transformation processes domain that we had: while designing $\mathcal{RS}^{PO^2}$, we already knew what was interesting, and what was not, and thus directly selected the parts of interest. However, as introduced in this section, the choices that have been made are not generic enough and cannot be replicated for any ontology. In the next sections we will detail this particularities and how to deal with the challenges and difficulties they raise.

### 2.4.1 Determination of explaining and explained attributes

Given how a transformation process is defined, we have imposed a learning constraint over the participants' and observations' attributes: if a probabilistic relation is found between them, the direction of such relation has to be from the participant's to the observation's attributes. Placing ourselves from a causal and explanatory view point, we can say that "*the participant's attribute **explains** the observation's attribute.*" Thus we determine that two attributes sharing a causal links can be defined as follow: the one causing is denoted the **explaining attribute**, while the one who is caused is denoted the **explained attribute**.

In a broader context, as long as we keep this causal and explanatory viewpoint, explaining and explained attributes are also present. Being able to identify them would allow us to set new learning constraints that could be translated into a relational schema.

**Example 11.** Be {A,B,C} a set of explaining attributes and {D,E,F} a set of explained attributes. This means that:

- any relation found inside the set is not structurally constrained: we have no *a priori* on their orientation.
- any relation found between attributes of the two sets is constrained: its orientation has to be from the explaining to the explained set.

This raises the question of the attribute identification, which finds an echo in Pearl's ladder of causality presented in Section 1.2. Indeed, if finding correlations in a dataset is easy, determining causation in the same dataset would however require external input. The first question we ask then is: would it be possible to determine if an attribute is either explaining or explained just from the given ontology, without expert knowledge? In other words, does the ontology own enough causal information to be able to dispense ourselves with humans interventions?

First of all, as mentioned in Section 1.1.7, some works tend to assume that ontologies' object properties are causal: in this case, determining explaining from explained attributes would be easy, as we could do the parallel between $range$ and explaining on one hand, and $domain$ and explained on the other. However, looking at most of the real-life ontologies shows that it is not as easy, since object properties are usually defined on anti-causal terms. For instance in PO$^2$, the relation $<$ observation, po2:observes, participant $>$ would bear no sense on a causal ground: the observation does not have to decide the state of the object of study! Moreover, some ontologies use **symmetric properties**, which are the inverse of the properties they reflect. In our example, it could be the relation <participant, po2:isObservedBy, observation> (which here would be causal, but does not exist in PO$^2$). In the case of such symmetric properties, how could we determine the

context of the attributes?

As a consequence an automatic detection would not be easily applicable, due to the variety of the ontologies defined. That is why in the next chapter, in addition of ontological knowledge, we introduce **expert knowledge**, the equivalent of human intervention.

### 2.4.2 Defining the temporality

The same way as the explanatory's constraints are given by the participants and observations, the temporal constraints are given by the succession of steps and the *po2:isBefore* properties. However, if causality between attributes can sometimes be difficult to define (meaning given two attributes we cannot always determine a constraint of direction for their probabilistic link), temporality is much more easier to define. Indeed, if we can assign to each attribute a certain time, then we can be sure that past events can always explain future one, without ambiguity.

If the representation of time is usually well controlled in ontologies (with hand-made properties such as *po2:isBefore* or by using universal ontologies such as the Time Ontology[1]), the way to achieve it is not democratized and can vary. This renders the elaboration of a generic algorithm hard, since we cannot provide a solution for every possibility in advance.

Moreover, not all ontologies use temporality to define their domain. The fact that PO$^2$ did eased our elaboration of ON2PRM since it gave us a first constraint easy to determine and understand: a step à time $t$ can only have an influence over steps at time $t+k$. In the case of an ontology not defined with temporality, this constraint has to be removed from the relational schema.

## 2.5 Conclusion

In this chapter, we have demonstrated that learning a domain described by an ontology is more precise if we also use the knowledge encompassed in the ontology. More precisely, we have presented a specific ontology, PO$^2$, dedicated to represent transformation processes. Using its semantic and our understanding of the domain, we have defined $\mathcal{RS}^{PO^2}$, a generic relational schema able to guide the learning of a probabilistic relational model from any domain ontology using PO$^2$ as a core ontology. The learning itself is done thanks to the algorithm ON2PRM that we have defined.

However, as raised in the previous section, PO$^2$ presents characteristics that cannot be found in all other ontologies. As a consequence the relational schema $\mathcal{RS}^{PO^2}$ and our algorithm are not generic enough to be applied elsewhere. In the scope of this thesis, it felt interesting to explore this path and try to see how our approach could be expanded to address any ontology.

[1] https://www.w3.org/TR/owl-time/

# CHAPTER 3

# INTERACTIVE BUILDING OF A RELATIONAL SCHEMA FROM ANY KNOWLEDGE BASE

**Contributions.**

Munch M., Wuillemin PH., Dibie J., Manfredotti C., Allard T., Buchin S. and Guichard E. Identifying control Parameters in Cheese Fabrication Process Using Precedence Constraints. *In:* DS 2018, Chypre.

Munch M., Dibie J., Wuillemin PH. and Manfredotti C. Towards Interactive Causal Relation Discovery Driven by an Ontology. *In:* FLAIRS 2019, Florida, USA.

Chapter 2 introduces our algorithm ON2PRM that allows to learn a probabilistic relational model from any domain ontology using the PO$^2$ ontology's core. The main idea is to use a relational schema in order to compartmentalize the different variables, and then to learn a probabilistic relational model. This method however is not generic, as it requires a pre-made relational schema tailored for PO$^2$. In the discussion, we presented the issues that were preventing us to apply it to any ontology, and concluded that human intervention was required to bring expert knowledge in order to explain some points. In this chapter, we will present how and to which extent this human intervention is needed, and how to handle it in order to semi-automatically build a relational schema from any ontologies. We will conclude by showing how this human intervention allows causal discovery.

**Section 3.1** continues the discussion initiated in the last chapter: we first come back to the different types of constraints and explain them (3.1.1); then we present the Stack Model $\mathcal{RS}$, a structure able to guide the construction of the relational schema for any ontology in order to

express these constraints (3.1.2).

**Section 3.2** presents CAROLL (Causal Assumption to pRobabilistic RelatiOnaL modeL), our algorithm of construction of $\mathcal{RS}$ that integrates expert knowledge. This is done in 4 steps: (1) defining an expert assumption (3.2.1); (2) selecting the corresponding variables (3.2.2); (3) enriching the learning dataset (3.2.3); (4) validating the learned model (3.3.1).

**Section 3.3** describes our method to deduce causal knowledge from the learned result. We first present how to validate the causality of arcs in the scope or our work (3.3.1). We then present the different possible conclusions we can draw (3.3.2), and present the particular case of incompatibility between our constraints' sets (3.3.3). Finally, we discuss the setting this causal discovery requires (3.3.4).

**Section 3.4** shows how we evaluate our method. It presents several examples taken from diverse domains, with different aims. First, we show that our algorithm is able to match our previous results (see Section 2.3) with a synthetic dataset reproducing a transformation process (3.4.1). In a second time, we show that it is able to deal with well-known knowledge bases with a portion extracted from DBPedia about movies, and how any expert can integrate its own knowledge in order to constraint the learning (3.4.2). Finally, we show that our algorithm is compatible with a real-life example on cheese processing with experts from the french National Research Institute of Agronomy (INRA) (3.4.3).

**Section 3.5** discusses the limits of our algorithm and Section 3.6 concludes this chapter.

**Experts, users and ontologies**

In the following, we will use the terms *expert* and *user* interchangeably. By definition, experts are knowledgeable both on the domain of interest and on the ontology's concepts themselves, while users are simply using our algorithm, following the steps described in section 3.2. However, in order to obtain interesting results, users have to be experts of the studied domain, otherwise they won't be able to provide interesting expert knowledge. In the following, we suppose we interact with an expert who wants to discover new facts about their domain.

## 3.1 Definition of a generic relational schema

Ontologies are built to encompass expert knowledge and structure it in a way that can be easily handled by computers. The first step of ontologies' design is to determine **competency questions** that allow to define their utility scope (Grüninger and Fox, 1995). To do so, expert knowledge

is required, usually under the form of human expert inputs. If the design choices they bring are useful to build an ontology close to the target domain, they however lead to design choices that can broadly vary depending on the experts and the competency questions. This explains in part the wide diversity we can encounter for ontologies. As a consequence, the idea of designing a fully automated passage from knowledge bases to probabilistic models is hard to conceive.

On another hand, they already have been ideas to propose the integration of human expert inputs during the construction of probabilistic of causal models. In the medical domain Jeon and Ko (2007) proposed a semi-automatic algorithm which extracted nodes from an ontology and let the expert draw the causal relationships between them. However, this method can be tiresome if there are a lot of nodes. Moreover the expert does not always know all the causal relations involved in the domain. On another hand, the approach proposed by (Devitt et al., 2006) also strongly depends on expert knowledge, while trying to lighten their work by integrating structuration from the ontology. Their approach consists of four steps: (1) selection of the variable of interest, (2) definition of their different values, (3) definition of the relations between the variables using the ontology's properties and (4) estimation of the conditional probability tables. This allows the expert to have a solid input over the Bayesian network construction, but also to rely on what the ontology already knows. However, this approach presents the pitfalls already mentioned in Sections 1.1.7 and 3.4.3, such as the fact that ontologies properties are not always causal by nature and not all have a structure that can be directly translated into a Bayesian networks.

The interest of our approach with probabilistic relational model is that it allows a two-times approach: first we build a global relational schema, that allows more liberty and imprecisions to the expert than a Bayesian network thanks to the classes compartmentalization and the relational slots. Once it is defined, it can also adapt to changes from the data: for instance, new experiments can be integrated, without having to modify the defined relational schema. In the next sections, we will present a template to easily build this generic relational model: the **Stack Model**.

### 3.1.1 Explicitation of constraints

When we built $\mathcal{RS}^{PO^2}$ we defined two types of constraints: temporal and causal. This distinction was done due to the specificities of the transformation process' domain, and the two types were treated the same way in the relational schema: a relational chain separating two classes (as shown in Figure 2.6). When building our new truly generic relational schema $\mathcal{RS}$, we wanted to keep this simplicity, and decided to keep the relational chain between the different classes to create constraints during the learning.

Thus, the same way as temporal and causal constraints were defined to keep track of the origin

of our learning, integrating expert knowledge in our generic relational schema requires that we distinguish between:

- **Ontology's structural constraints**, which are relational slots directly derived from the ontology and whose causality have been approved by the expert.
- **Expert's causal constraints**, which are relational slots defined by the expert in addition of the one he has validated from the ontology.

As we will see in the next section, the ontological structure and the expert's constraints both lead to defining relational slots. Yet, it is important to keep in mind that even if a constraint has been directly derived from the ontology, it **always** has to be approved by an expert. Moreover, it is also possible that the expert rebut all ontology's semantic constraints, if they deem them not causal: the relational schema would be in that case entirely defined by the expert. As a consequence, the distinction we introduce here is only to show that the expert will have the possibility of keeping part of the ontology's structure (such as the temporal properties) to guide the construction of the relational schema. **In the following, we keep the distinction in order to show whether a design choice has been decided considering the ontology original's semantic or the expert's decision.**

The main difference between our model and the others presented in the state of the art is that we never assume a relation between two attributes. The relational schema we build with the ontology and the expert only tells us that "*This attribute **can explain** this other attribute*". We decide the direction, but not the presence of the relation, which itself depends on the data encompassed in the ontology.

**Definition 7:** *Potential explanation*. Given two variables $A$ and $B$, and a learning constraint $C_{A \to B}$ such that if a probabilistic link is found between $A$ and $B$, it has to be oriented from $A$ to $B$. We say that $C_{A \to B}$ is a **potential explanation**. $A$ **potentially explains** $B$, and $B$ **is potentially explained by** $A$.

If a probabilistic relation is found between $A$ and $B$, their potential explanation can be translated into two kinds of relations: (1) a direct probabilistic relation from $A$ to $B$ or (2) a sequence of probabilistic relations from $A$ to $B$ such that, given other variables $X$, $Y$, ... it exists a path $A \to X \to Y \to ... \to B$. This distinction is important, and will be covered in more details in Section 3.3. For now, we consider that a **causal path** from $A$ to $B$ can be structured following one of these two examples.

### 3.1.2 Structure of the Stack Model

In the same way we used $\mathcal{RS}^{PO^2}$ in the last chapter to guide the construction of a relational schema dedicated to a specific transformation process, we now aim to define a guideline $\mathcal{RS}$ for the construction of a relational schema for any ontology. To do so, we introduce the **Stack Model**, which is a guideline to build relational schemas. This appellation comes from the fact that in all the relational schemas we will define using it, classes are piled together. Their relational chains go from top to bottom, allowing the causal relations to *"flow"* from the higher classes to the lower. This structure encompasses the two kinds of constraints we introduced before, ontological and expert. In the next section we will illustrate how these constraints are introduced in order to build a specific relational schema.

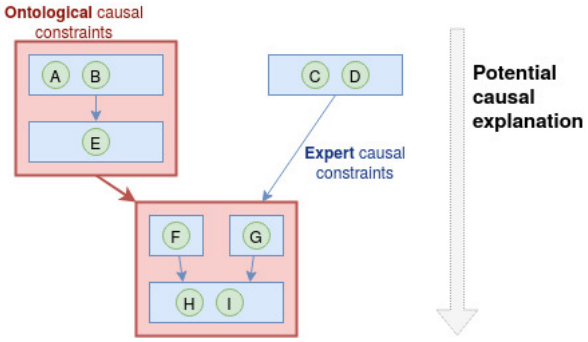



***Figure 3.1:* Structure of the Stack Model $\mathcal{RS}$.** This generic relational schema can adapt to any ontology. During its building, it can encompass between ontological (red) and expert (blue) constraints.

A Stack Model is composed of different classes, that can encompass other classes themselves, in the same way than a Step class in $\mathcal{RS}^{PO^2}$ was composed of the Participant and Observation classes. These classes own attributes whose relations with other attributes are influenced by their position in the Stack Model. Indeed, attributes in the upper class are considered as *potentially explaining* the attributes in the lower classes, while attributes from the lower classes are considered as *potentially explained* by those of the higher. On another hand, if two attributes are from the same class, then the orientation is impossible to determine from the given constraints alone. Finally, in the same way there had to be a slot chain between steps for their attributes to share a probabilistic link in $\mathcal{RS}^{PO^2}$, there must be a slot chain between the classes for their attributes to interact.

**Example 12.** Given the example described in Figure 3.1:

- $A$ potentially explains $E$ thanks to an expert causal constraint.
- $A$ potentially explains $F$ thanks to an ontological causal constraint.
- $A$ and $C$ cannot have any probabilistic relation because there is no slot chain between them. The same can be said on $F$ and $G$, since they are not in the same class: the

subdivision made by the expert causal knowledge prevents them of sharing a slot chain.

- $A$ and $B$ on the contrary are in the same class: they can share a probabilistic link, but we cannot assess its orientation from the given constraints alone.

### 3.2 CAROLL Algorithm

In this section we present the CAROLL (Causal Assumption to pRobabilistic RelatiOnaL modeL) algorithm, whose overall overview is given in Figure 3.2. On the contrary of the ON2PRM algorithm, we are here agnostic when considering the domain we want to model. As a consequence, before starting to build a relational schema, we need a motivation. For PO$^2$, the motivation was to understand how each participant and their values were influencing the observations and the final result of the transformation process. In our case, we need an expert to give us this motivation, which is formulated as an **expert assumption**: a causal statement about the domain the expert wants to verify (**initialization**). From there, a first relational schema is semi-automatically built (**step 1**), that is then enriched in a second time (**step 2**). Using this relational schema, we learn a model and submit it to the expert that can validate it or not (**step 3**). In case of reject from the expert, the relational schema has to be built again, or the model is considered as incapable of answering the expert's assumption.

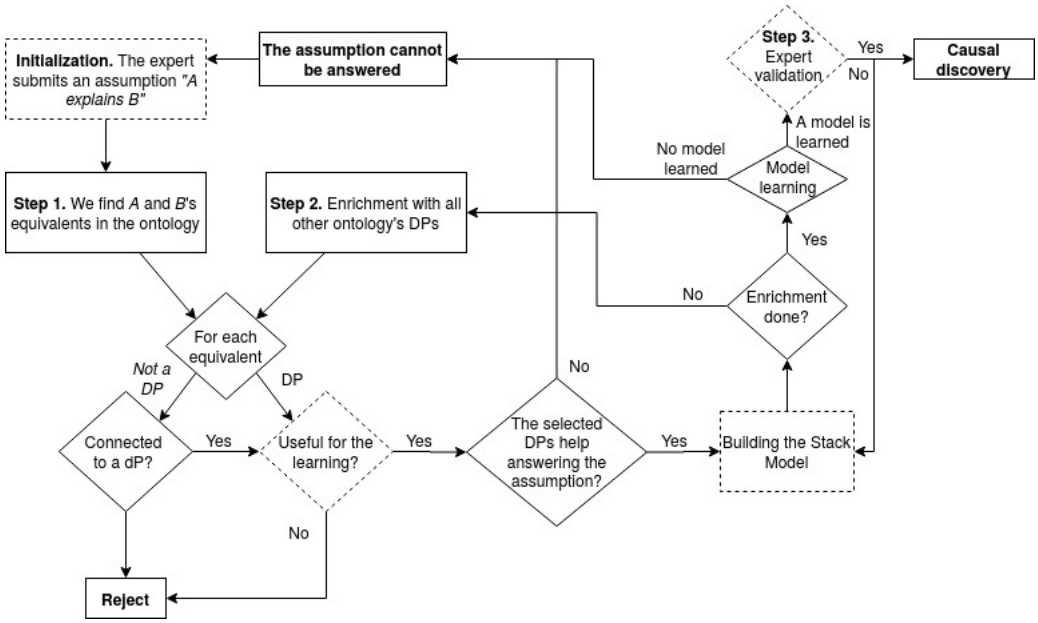



*Figure 3.2:* **Overview of the CAROLL algorithm.** This algorithm is composed of three different steps. Each dotted line indicates when a human expert's input is required.

In the rest of this section we cover the different steps and how the expert knowledge intervenes. As an illustration, we consider the small ontology about university shown in Figure 3.3.

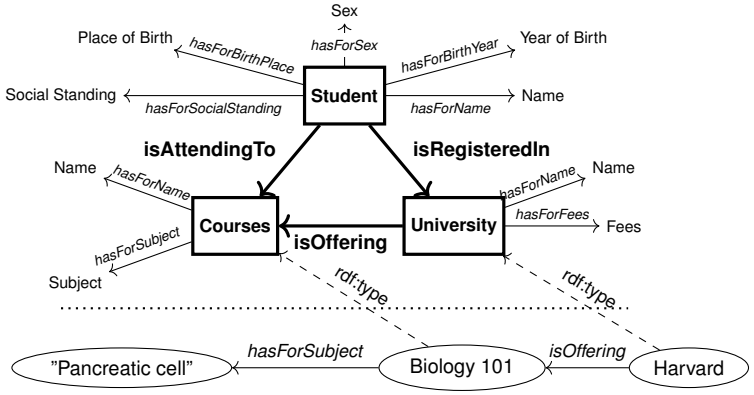



*Figure 3.3:* **Excerpt of a knowledge base about students and universities.**

This ontology is composed of three main $O$-classes (*Student*, *Course* and *University*), each being characterized by specific datatype properties.

### 3.2.1 Expert assumption

As we have defined in Section 2.2.1, a good statistical learning relies on three criteria: (1) presence of discret data, (2) this data is comparable and (3) it is instantiated enough. In order to apply the method described in the next section, these pre-requisites must also be met: the knowledge base we wish to study must have datatype properties that are instantiated enough and are linked together through associations of classes instances and object properties. However, expanding our methods to knowledge bases other than PO$^2$ raises a new question, which is **whether a given knowledge base can be translated into probabilistic models or not**. To answer this question, the expert must determine if the study of this knowledge base can help him to better understand the domain: they must have a **goal**.

In order to initialize our algorithm, we need a goal for the model we want to learn. This goal takes the form of an expert's causal assumption and serves two purposes: (1) it gives a criteria to estimate the quality of the model: *"Can the learned model validate or not this assumption?"*. The same way as ontologies' competency questions determines the scope of an ontology, the expert assumption gives a purpose to the model learning. Moreover, it (2) helps the user to make their selection and estimate if a variable is interesting or not.

**Definition 8:** *User's causal assumption*. Be $\mathcal{H}$ a user's causal assumption, and $\forall i \in \{1, n\}$ $\forall j \in \{1, m\}$, $C_i$ and $E_j$ variables of a domain. We express $\mathcal{H}$ as *"$C_1$, $C_2$,... $C_n$ have a causal influence over $E_1$, $E_2$,... $E_m$"*, and denote

- $C_1$, $C_2$,... $C_n$ as the explaining attributes.
- $E_1$, $E_2$,... $E_m$ as the explained attributes.

Since the user's assumption is supposed to transcribe a causal reasoning, we will sometimes present the explaining attributes as **causes** and the explained attributes as **consequences**.

> **Example 13.** As an illustration, using the university ontology, we pose $\mathcal{H}_U$: *"The birthplace of a student has a causal influence over its university"*.

The user's assumption $\mathcal{H}$ is a statement the expert wishes to verify through the analysis of the learned model. It transcribes a potential causality between the attributes $C_i$ and $E_j$: the expert does not know if there is a relation, but if there is one, then it is oriented from $C_i$ to $E_j$. However, the presence of a path alone is not enough to determine wither it is causal or not: it also need to be causally validated, through methods we will detail in the following. As for now, we define a **causal path** as a path (direct or indirect) between two variables whose causality has been validated. Once the model is learned and **validated**, several cases are possible:

1. Causal paths are indeed found from all the explaining attributes to all the explained attributes: $\mathcal{H}$ is totally verified.
2. Causal paths are found from some of the explaining attributes towards some of the explained attributes: $\mathcal{H}$ is partially verified.
3. No causal path is learned between the explaining and explained attributes: $\mathcal{H}$ is refuted.

It is important to note that the absence of causal path between $C$ and $E$ variables is not a way of evaluating the learned model, as it is only a possible result. However, any of these possible results can only be concluded if the model has been validated by the expert beforehand. We will detail the protocol of validation in Section 3.4.

If the assumption gives a statement that the user wishes to verify thanks to the learned model, then the so-called model has to own variables that would represent the main attributes of the assumption. This raises a new interrogation: with PO$^2$, we considered only the participant's and observations *po2:hasForValue* datatype property as able to define new attributes. However, it was a specificity of this particular ontology, that not all ontologies share. As a consequence, we need to expand our definition of potential variables for building the relational schema.

Since the learning is based on statistical counting, we need values that can be repeated and compared. For this, we chose to keep looking at datatype properties, which are the only way to associate datatypes (i.e. values) to ontology concepts. As a consequence, we consider by default all datatype properties as potential variables for the relational schema (and thus do not restrict this definition to only one type of datatype property as it was the case in PO$^2$). However, this opening raises new issues, as not all datatype properties are considered **useful for the learning**, for several criteria:

**C1.** The datatype property is not relevant to the problem, meaning it cannot help to answer the assumption. This is a subjective reason, that can only be raised by the user. This is due to the fact that ontologies can sometimes be very broad and cover a huge part of a same domain. In this case, the user reserves the right not to include it in the relational schema.

**C2.** The datatype property cannot be used for statistical learning. This is the case when the values it takes: (1) are all the same (e.g the characteristic of an apparel that is used for all repetitions) or (2) are all different (e.g. an ID). In this case, the statistical learning is impossible, since the datatype property's values do not seem to be influenced by other variables in the model.

**C3.** The datatype property has not enough instantiations. This is also linked to the fact that we wish to learn using statistical methods: if we do not have enough repetitions, then using this datatype property would bring a lot of missing values, which would render the learning imprecise and lower its overall quality.

**C4.** The datatype properties' values cannot be discretized. It is especially important in case of a continuous set of values (e.g. an income, a population size), but has to be considered as well in case of finite set of values with too many different values (e.g. the different cities of a country). In those cases, the statistical learning cannot be directly done, as we need to discretize in order to create categories. The number of categories has to stay acceptable in regards of statistical learning standards: the more they are, the more difficult the learning will be. This discretization has to be decided by the expert, as its design can greatly influence the result of the learning. If however the expert cannot define a proper discretization then the datatype property has to be discarded.

Using these restriction, we pose the definition of the attributes useful for the learning, to which we will refer in the following.

> **Definition 9:** *Attribute useful for the learning*. An attribute useful for the learning is a datatype property whose set of values does not present one of the criteria C1, C2, C3 or C4.

### 3.2.2 Assumption's Attributes Identification

This part is done in several steps: (1) recovery of the datatype properties corresponding to the assumption's attributes; (2) verification of their connections with each other and (3) their ability to answer the assumption. If all these steps are validated, we create the Explaining and Explained classes in the relational schema.

From the user's assumption, a first selection is made (as shown in **Step 1** of Figure 3.2). This selection is based on a similarity analysis of the attributes composing $\mathcal{H}$ and looks for their equiv-

alent in the ontology. The goal of this selection is to gather variables for the relational schema that would represent the attributes defined in $\mathcal{H}$.

For all variables $C_i$ and $E_j$ in $\mathcal{H}$, we look for their equivalent in the ontology using a similarity measure (such as the Jaccard's measure we selected for our work). To do so, we go through all the knowledge base's entities and compute the similarity measure between the variable name and the entity's *rdfs:label* (i.e. its name). If the measure is higher than a tolerance threshold fixed by the user, then the entity is kept and treated differently depending of its type:

**T1** **Datatype property:** if the entity is a datatype property that respects the criteria to be an attribute useful for the learning, then it is kept.

**T2** **$O$-class:** if the entity is an $O$-class, then we look at all its datatypes properties. For each we apply T1.

**T3** **Object Property:** if the entity is an object property, then we look at the classes at its range and domain and apply to each T2.

**Example 14.** Given $\mathcal{H}_U$: *"The **birthplace of a student** has a causal influence overs its **university**"* and the university ontology, we define two attributes:

- **birthplace of a student**: we select the datatype property *hasForBirthplace*
- **university**: we select the $O$-class *University*, which owns two datatype properties: *hasForName* and *hasForFees*.

We consider that *hasForBirthplace* and *hasForFees* as useful for the learning, and keep them. However, *hasForName* (which refers to the university's name) does not respect R4: for the sake of the example, we consider that the number of universities' names is too important and that it is not possible to discretize them. As a consequence, *hasForName* is discarded.

Once the set of the datatype properties useful for the learning and corresponding to the assumption have been selected, we realize a first verification to see if they are "connected". Indeed, the interest of learning a probabilistic model is to verify the evolution of each variable in correspondence with the others: as a consequence, we have to be able to tell, for each example, the value that every variable takes relative to the others. This requires that each datatype property is joined to the other *via* a path in the knowledge base, i.e. a combination of object properties that links the $O$-classes domains of the selected datatype properties. If such a path exists, then we have a connected knowledge graph. Otherwise, the attributes that cannot be connected are discarded.

When the final set of selected datatype properties has been built, we have to verify if it can answer the assumption, i.e. if the set is enough to represent each attribute of the assumption. If not,

we consider that the knowledge base cannot answer the assumption and we stop the algorithm: the assumption has to be changed. To do so, we review each variable of $\mathcal{H}$ and check its set of attributed datatype properties:

- If there is none, this variable cannot be represented considering the information encompassed in the knowledge base: the assumption **cannot be answered**.
- If there is at least one, this variable is **validated** and we look at the others.

If all variables are validated, then the knowledge base is sufficient to answer the assumption. We create two classes in the relational schema, *Explaining* and *Explained*, in which we sort the corresponding attributes. The Explaining class is placed on the top, meaning that it is potentially causal of all the variables in the class under it (and thus the Explained class as well). If required, the user can also add expert knowledge in order to stratify the variables inside the classes. This expert knowledge can take two forms:

- $C_1$ and $C_2$ have no interaction $\rightarrow$ they are placed in two parallel classes without relational chain
- $C_1$ potentially explains $C_2 \rightarrow$ we create two stacked classes and place $C_1$ in the upper one, $C_2$ in the lower.

**Example 15.** From $\mathcal{H}_U$, we trace the relational schema $\mathcal{RS}_U$ presented in figure 3.4.

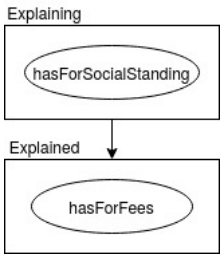



***Figure 3.4:*** **Relational schema $\mathcal{RS}_U$ built from $\mathcal{H}_U$.** From the user's assumption, we only have two attributes that have been placed in the Explaining and Explained classes.

Once the relational schema is built with the classes containing the attributes built from the selected datatype properties, the learning is theoretically possible. However, we only have created the attributes that are directly about the assumption: it is then highly probable than the knowledge base owns other potential attributes useful for the learning that would enrich the model. This enrichment step is described in the next section.

### 3.2.3 ENRICHMENT

The act of enrichment consists of adding new attributes to the relational schema's classes in order to have more information and precision about the possible causal relations. Indeed, in case of

indirect causation (i.e. when $A$ has a causal influence over $B$ through a causal path with other variables $X$, $Y$, ...), even if the main causal relation is learned ($A \rightarrow B$), the final graph lacks of explanation (we miss the fact that $A$ influences $B$ through $X$ and $Y$). Since the user's assumption only covers the most important part of the model, we need to complete it so we get the most accurate possible answer.

In order to enrich the model, we need to find other attributes useful for the learning, and integrate them in the relational schema. To do so, we must assure that they are linked together, the same way as the selected attributes were joined by a chain of object properties. As a consequence, we find new attributes to enrich our model by radiating around those we already have found in the knowledge base. For each entity, we look at the triples it is involved in and the entities that share these triples. These entities are also analyzed, until every potential attribute useful for the learning has been found.

1. For each datatype property selected as an attribute useful for the learning, we look at its domain $O$-class.
2. For each $O$-class, we look at its datatype properties and check if they are useful: if so, they are added in the relational schema.
3. For each object property linking the $O$-classes we are studying, we look at the other $O$-classes and study them too.

**Example 16.** From the University ontology and $\mathcal{H}_U$, we also select *hasFor Sex*, *hasForBirthPlace* and *hasForSocialStanding* to describe the student; and *hasForSubject* to describe the courses he is enrolled in.

In the end, we gather a set of datatype properties, that represent a group of variables we want to place in the relational schema. Unlike the previous section were the variables were placed automatically (in the Explaining and Explained classes), this placement is done by the expert, according to its knowledge of the domain. They have the ability to create new classes they can place above, under, or in parallel to other classes. In some cases, they can also use the ontology structure to define automatic classes in the relational schema (for instance, using the *po2:isBefore*). In this case, they are given the possibility of creating as many classes as there are of instances of the $O$-classes, which are automatically linked by relational slots mimicking the selected object property (in our example, *po2:isBefore*). However, we do not develop this particular in this chapter, as it echoes our following work and raises new problem about automation that we do not see fit to present for now. For a more detailed approach on how to automatically deduce a relational schema from a knowledge base's semantic, see Chapter 4.

**Example 17.** We update $\mathcal{RS}_U$ with the new variables that we have found. We estimate that the birthplace, the birth date and the sex can potentially explain the social standing and create Class 0. On another hand, we estimate that the information about the courses are potentially explained by the university, but not the contrary: as a consequence, Class 1, which represents the informations about the courses, is placed under the Explained class. This means that even if Class 1 can bring more information about the model, it is not required to answer $\mathcal{H}_U$. The result is shown in Figure 3.5.

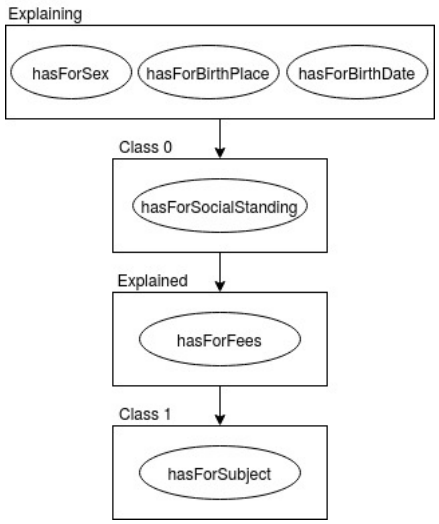



***Figure 3.5:* Relational schema $\mathcal{RS}_U$ built from $\mathcal{H}_U$ updated after enrichment.** The new Class 0 and Class 1 classes have been manually added by the expert.

### 3.2.4 Validation

Once the relational schema has been fully designed, we can learn the relational model following the same protocol as presented in the previous chapter with $\mathcal{RS}^{PO^2}$. For each class of the relational schema, we build a dataset in order to learn a small Bayesian network, which will be the base for the class's relational model's structure. However, this dataset is built by taking into account attributes from the class we aim to represent in the relational model, but also from other potentially causal classes (for instance, the relational schema of Figure 3.5 shows that in order to learn **Class0**, we also need to include in the dataset attributes from **Explaining**). From there, it becomes possible then to learn probabilistic relations between attributes that are both not part of the class we aim to represent. As a consequence, for Bayesian network learned from each class, we only keep the probabilistic relations learned that include at least one attribute of the considered step (see Section 2.2.3). All the other relations are not kept in the relational model.

**Example 18.** If we consider the relational schema defined in Figure 3.5:

- The **Explaining** class is learned with the following dataset:

| Student | student.hasForSex | student.hasForBirthPlace | student.hasForBirthDate |
|---|---|---|---|
| s1 | s1.sex | s1.birthPlace | s1.birthDate |
| ... | ... | ... | ... |

  Since it is the first class with no class parent, all learned relations are kept.

- The **Class0** class is learned with the following dataset:

| Student | student.hasForSocialStanding | |
|---|---|---|
| s1 | s1.socialStanding | **Explaining class dataset** |
| ... | ... | |

  However, only the relations involving the social standing are kept.

- The **Explained** class is learned with the following dataset:

| Student | student.hasForUniversity | university.hasForFees | |
|---|---|---|---|
| s1 | u1 | u1.fees | **Class0 class dataset** |
| ... | ... | ... | |

  Once again, all the relations involving the university fees are kept. In this dataset, it is important to note that we introduce the *student.hasForUniversity* column, which forces to look only at the university in which the student is: this highlights the importance of the connected data, i.e. each instance has to be linked to the others in order to compare them.

The system is then instantiated and the resulting model (a Bayesian network) is then presented to the expert.

A first important thing to note is that a Bayesian network does not automatically allows causal discovery, as it was covered in Section 1.2. In order to analyse the result and determine if it is valid or not, we need first to apply methods in order to discover causality and determine if for all arcs' orientations, those are causal or not. These methods are detailed in the next section. Once done, the model we have learned is thus partially or completely causally oriented. Two verifications have then to be done:

1. All the causally oriented arcs are coherent for the expert.
2. There either is a causally oriented path from the explaining attributes (or some of) to the explained attributes (or some of), either no path at all.

If and only if those two conditions can be verified, then the causal assumption can be answered. However, one must keep in mind that this model was learned under two strong assumptions: (1) the causal knowledge brought by the expert is true and (2) the data described in the knowledge graph represents the reality (and not just a part of it). False information can indeed lead towards the learning of a model that does not represent the reality, but a distorted one we assume was true. A better discussion of the consequences of such a learning are given in Section

3.3.4.

As a consequence, even if the model cannot answer the assumption, it helps raising new questions for the expert. In the following we present the classical issues an expert can encounter during the validation and how to deal with them.

- **A relation is learned but shouldn't exist.**
  - **The orientation is wrong.** It means that the relational schema has to be questioned.
  - **The relation itself has to be questioned.** It means that the knowledge base is biased and not balanced. New experiments should be done under other conditions and added for the model learning.
- **A relation is not oriented.** The relational schema and knowledge base alone are not enough to determine the causal orientation of the relation. The expert should run interventional experiments (if possible) to determine the orientation.
- **A relation should exist but is not present.** The knowledge base is biased and does not present all the possibilities. The expert should do more experiments to test the relation between the two variables in questions and add their results to the knowledge base.

As a consequence, even if the assumption is not answered, the model can suggest new experiments and modifications that challenge the expert's knowledge in order to give a better overview of the domain.

## 3.3 Towards causal discovery

Once the model learned and instantiated, we have a Bayesian network whose relations are oriented. However, before causal validation or rebuttal, we are not able to deduce if these orientations are causal or not. In this section, we will cover a method able to help the expert validate the causal orientation of the arcs. Once this work done, it can be presented to the expert so they are able to validate or rebut the model, as described in the previous part. We will also discuss its limits and the precise framework in which the causal discovery is possible.

### 3.3.1 Validating causal arcs

Causal discovery is possible in our particular case because our model is learned under causal constraints given by the potentially causal relations. These relations are all encompassed in the relational schema and thus are directly transcribed in the learning, which results in a model *influenced* by the ontological and expert knowledge. This result can be seen as the intersection of all the model constrained by the datatype properties' values on one hand and all the model constrained

by the expert knowledge on another. This reduces the search space, and orients our learning towards the true causal model.

However, the informations brought during the learning are usually not enough to determine the whole truth and, in the end, multiple models can be possible. Similarly to the essential graphs presented in Section 1.1.4, with this method we determine, among all the possible models, which ones have the same orientations. We consider these common arcs as causal.

In order to analyse these common arcs, we propose a protocol in three steps: first analyzing the inter-classes arcs in the relational schema, then complementing what we deduce with the essential graph. This protocol is represented in Figure 3.6.

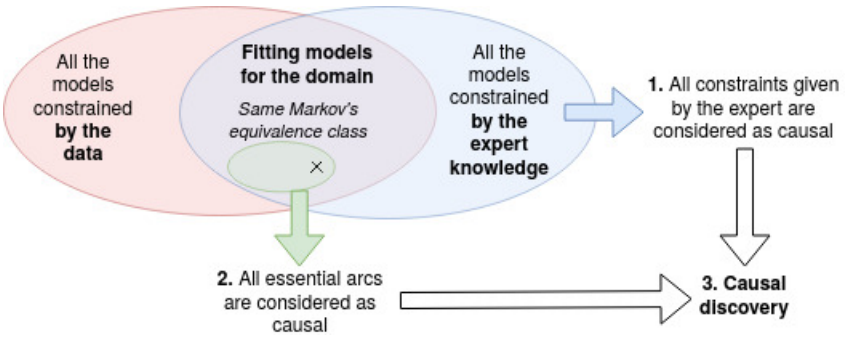



*Figure 3.6:* **Causal Discovery protocol when a model is learned.** The set on the left represents all the models that respect the constraints given by the data. The set on the right represents all the models that respect the expert knowledge's constraints encompassed in the relational schema. The set at the intersection represents all the models that respect both of these constraints. The × represents the learned model (i.e. the one that is the most probable given the set of constraints). The smallest set in the intersection represents ×'s Markov's equivalence class.

#### Causal arcs validated by constraints: the inter-classes relations

Since all classes have been determined either by the expert's causal assumption (Explaining and Explained class) or the expert himself (during the enrichment), we consider that if a relation follows a given direction, its orientation has been validated by the expert. As a consequence, we can automatically consider all inter-classes relations as causal given the information brought by the expert.

During the validation, those relations are the first to be submitted to the expert, since they are a direct result from their input in the CAROLL algorithm. If they do not agree with a conclusion, then it means that the relational schema itself is wrong and should be re-evaluated.

#### Causal arcs validated by the essential graph: the intra-classes relations

On the contrary of inter-classes relations, intra-classes relations have not been directly influenced by the expert. By essence, all variables in a same class are variables to which the expert could

| Case | Type of relation | Essential Graph | Expert Validation | Knowledge base Validation | Causality |
|---|---|---|---|---|---|
| 1 | Inter-class | $\rightarrow$ | True | True | **Validated** |
| 2 | Inter-class | — | True | False | **Validated** |
| 3 | Intra-class | $\rightarrow$ | False | True | **Validated** |
| 4 | Intra-class | — | False | False | **Not Validated** |

***Table 3.1:*** **Different cases for causal validation.** Depending of the type of the relation and its orientation in the essential graph, different cases are possible. As stated before, these conclusions are only possible if the (1) the causal knowledge brought by the expert is true and (2) the knowledge base's data represents the reality.

not bring causal order, usually because he did not have information. As a consequence, more variables there are in a class, more probable it is to have intra-classes relations. Their orientation is not fixed by expert knowledge, and in their case it is possible that the learned model does not represent the reality.

In order to determine which can be oriented, we look at the essential graph. As defined in Section 1.1.4, the essential graph is a semi-oriented graph that shares the same structure as the considered Bayesian network, but whose arcs' orientation transcribes essential arcs (i.e. arcs always oriented the same way in the Bayesian networks' Markov's equivalence class). In the introduction of this section, we presented our learning as the result of an intersection between two sources: the data brought by the ontology and the expert knowledge. As shown in Figure 3.6, the result of our learning is at the intersection of the two constraints obtained from these sources, and it is also part of a Markov's equivalence class (shown in green). This means that when consulting the essential graph of the learned model, we have access to all the graphs that respect the expert's constraints. Under this pre-requisite, we then determine that if an arc is an essential arc (i.e. it is always oriented the same way), then it is considered as causal (i.e. all the models that respect the set of constraints present this orientation).

### 3.3.2 Possible conclusions

It is important to note that these two methods are complementary, and cannot lead to contradictory results. This is due to the fact that the expert knowledge guide the learning, and automatically rule out models whose essential graph could contradict it: in other words, it is impossible to learn an essential graph's oriented arc that would go against the potential causality defined by the expert. Considering this, all possible cases are described in Table 3.1. Moreover, in the last case, it is sometimes possible to deduce the causality by propagation of the other causal constraints we deduced and validated.

**Example 19.** Consider three variables $A$, $B$ and $C$ such that $A \rightarrow B \rightarrow C$. When learning the model, since there are no V-structure, we thus obtain the following essential graph $A—B—C$,

meaning that either (i) $A \rightarrow B \rightarrow C$ or (ii) $A \leftarrow B \leftarrow C$ are possible.

We suppose that the expert had put a potential causality between $A$ and the set of $B$ and $C$, thus creating two classes, one with $A$ and another with $B$ and $C$. As a consequence, when referring to the Table 3.1, we have two cases:

- Between $A$ and $B$, we have the case 2: the causality is validated by the expert knowledge, and even if the essential graph cannot help, we can infer that $A \rightarrow B$.
- Between $B$ and $C$, we have the case 4: neither the expert nor the essential graph can help.

However, as we have said, the essential graph pointed towards only two models, (i) and (ii). Given what we have deduced between $A$ and $B$, the only remaining possible model is thus (i): we can then infer $B \rightarrow C$.

Once all of the arcs that could be causally determined have been oriented, we can now evaluate the model. If the expert does not agree with a causal relation, the model has to be re-evaluated the way we presented before. On the contrary, if the model seems plausible, then we can look for causal path between the explaining and explained variables. However, if a path exists but has a non-causal relation in it, then it is not possible to deduce anything.

Three conclusions are possible for the study of the causal influence of explaining over explained attributes:

- Direct causation ($A \rightarrow B$). Given our university example, it would mean finding a direct causally validated relation between $hasForBirthPlace \rightarrow hasForFees$.
- Non-direct causation ($A \rightarrow ... \rightarrow B$). That would mean finding for instance a causally validated relation such that $hasForBirthPlace \rightarrow hasForSocialStanding \rightarrow hasForFees$.
- Independence (no directed path). Other possible paths could have been found, such as for instance $hasForSocialStanding \rightarrow hasForFees$. But there is no causally validated path between $hasForBithPlace$ and $hasForFees$.

### 3.3.3 INCOMPATIBILITY OF CONSTRAINTS

In some cases, it is possible that the two sets of constraints (expert's and knowledge base's) are incompatible, resulting in the absence of a learned model. If this particular situation clearly indicates that it is not possible to answer the expert causal assumption, it however brings some indications that can be used to do causal discovery.

Indeed, the same way as anti-causal orientations or presence of arcs between independent variables can indicate a problem either in the data set or in the expert knowledge, the absence of learned model strongly suggests that the expert should reconsider either their learning set or

what they know about the domain.

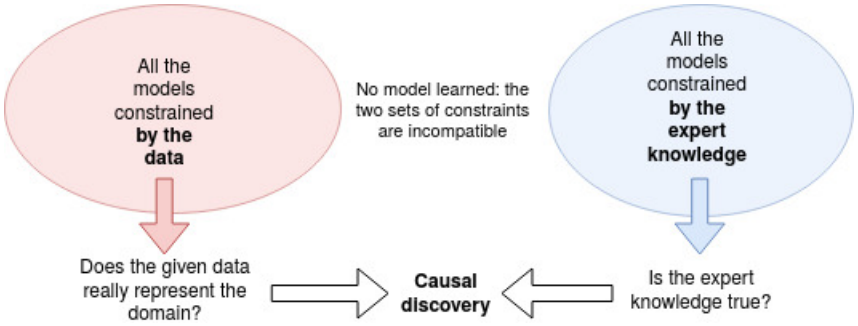



***Figure 3.7:*** **Causal Discovery protocol when a model cannot be learned.** This particular case can be explained by two reasons: either the dataset does not represent the whole domain, or the expert constraints are not correct.

As a consequence, the causal discovery can be done by answering two questions (as illustrated in Figure 3.7).

- *"Does the dataset really represent the domain?"* Maybe other entries should be added in order to be more generic.
- *"Are the expert constraints true?"* Maybe they are false and/or too restrictive.

### 3.3.4 Discussion

As we brushed before, this causal discovery is only possible because of particular pre-requisites that we distinguished in Section 1.2.2.

#### Causal Sufficiency

Causal sufficiency indicates that all variables needed to learn a complete causal model are taken into account. Indeed, a classic pitfall of causal discovery is to consider only a part of the important variables, which would lead to fallacious causal relations: if $A$ is a common cause to $B$ and $C$ but is omitted in the model, it is highly probable that a relation between $B$ and $C$ is learned, despite the absence of direct causation in the true model.

As a consequence, the relational schema built with CAROLL must take into account all relevant variables. Usually, the user's assumption is precise enough to avoid this issue, and the enrichment is there to help the expert to check within the ontology all the potential variables he could have omitted. Yet, if the knowledge base partially represents the domain, the causal sufficiency can be questioned.

#### Data quality

As we described in Section 1.2.2, multiple criteria define the data quality: missing data, selection bias, measurement error, non stationary or heterogeneous data and deterministic case. If one of them is not respected, then the data used does not represent the domain, and thus the possible results are biased.

#### Consequences

If one of these two pre-requisite is not respected, then the intersection that we studied in the previous section is not true anymore, and the assumption about the essential arcs being causal does not stand. As a consequence, all the causal discovery is invalided.

### 3.4 EVALUATION

In this section, we will present three applications of the CAROLL algorithm. The first application is similar to the one of the previous chapter, since once again it covers transformation processes. The interest of this part is to show that through a different algorithm we can obtain similar results. The second application shows how CAROLL can handle classical ontologies with an extract from DPPedia. In this part, we play the role of experts as we tackle the films' domain. Finally, the third application proposes a real life example with the analysis of a domain ontology used by experts of the National Research Institute of Agronomy for cheese processing.

#### 3.4.1 SYNTHETIC DATA SET

In this experiment, we present a simple example of a transformation process designed by us. Fig. 3.8 presents its instantiation in PO$^2$ and Fig. 3.9 (a) its model, built such that each attribute is represented by one binary variable. For simplification purposes, we suppose here that the original object properties *hasForParticipant* and *hasForObject* are directly datatype properties here, in order to simplify the illustration. This does not change the overall idea of PO$^2$. The same way we explained in Chapter 2, we generate from this model 5,000 different instances of the transformation process, which corresponds to a knowledge base of 165,000 RDF triplets. Since every variable is known and binary, we consider for the rest of this experiment all defined datatype properties as useful for the learning. The user's causal assumption $\mathcal{H}_f$ is: "$p_1$, $p_2$ *have an influence over* $o_4$, $o_5$".

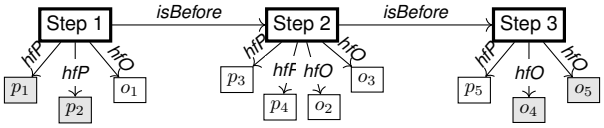



***Figure 3.8:*** **Example of an instantiated transformation process using the PO$^2$ ontology.** The terms *hfP* and *hfO* are respectively short for the datatype properties *hasForParticipant* and *hasForObservation*.

#### Relational Schema Construction

$\mathcal{H}_f$ defines two participants $p_1$ and $p_2$ considered as explaining, and two observations $o_4$ and $o_5$ considered as explained. For each we select the attribute useful for the learning corresponding in the knowledge base. For the enrichment, we use the expert's knowledge of both the domain and the ontology: all participants' useful datatype properties can be considered as explaining and all observation's useful datatype properties as consequence. Due to the temporality induced between the steps, we also introduce potential explanations between Step 1, Step 2 and Step 3. This result adds new classes in the initial relational schema: for each step $S_i$, we define one explaining class $E_i$ and one consequence class $C_i$, organized such that for all $i < j$, $E_i$ and $C_i$ classes can explain $E_j$ and $C_j$ classes. Fig. 3.9 (b) shows the essential graph learned and its relational schema.

#### Results Discussion

In this small example, the domain is known enough for the classes to be precise (i.e. the expert was able to give potential causal relations between nearly every variable). As a consequence, the final model 3.9 (b) only presents inter-classes relations (which are automatically causally validated by definition). It is interesting to note that a lot of these relations are also validated by the essential graph.

Since the generative model is known for this experiment (Fig. 3.9 (a)), we compare the result of learning using the relational schema (Figure 3.9 (b)) and without (Figure 3.9 (c)). In this example, the intake of the relational schema is important: while (b) is close to the real model (only one relation has not been learned), the model learned without relational schema (c) is not as close and even suggest one anti-causal relation (from $o_3$ to $p_4$). Moreover it leads to a false answer to $\mathcal{H}_f$: $p_2$ is shown with an indirect influence over $o_5$ through $p_4$, whereas it is not the case in the true model.

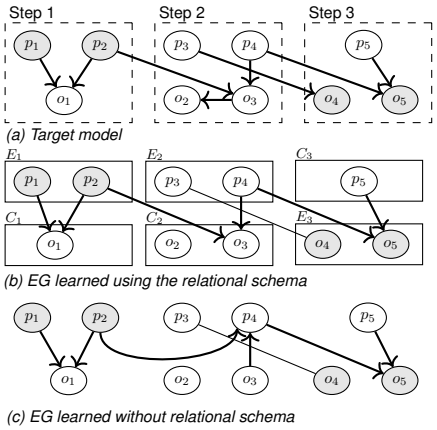



*Figure 3.9:* **Comparison of the different learnings.** (a) Ground truth. (b) Model learned with the CAROLL algorithm. (c) Model learned without the CAROLL algorithm. The grey variables are the expert's causal assumption's.

### 3.4.2 MOVIES

The DBpedia database collects and organize all available information from the Wikipedia[1] encyclopedia. Since it describes 4.58 million things (including persons, places, ...), we have decided for our test to only study a small part of it, on a subject simple enough where we could easily play the role of an expert. To validate our approach we use the DBPedia part dedicated to films and the user's causal assumption $\mathcal{H}_e$: *The **origin country** of a film has an influence on its **number of won awards***. We play the role of the user: since we are not expert of the domain, the results have to be taken lightly. However, this example allows us to show that (1) the CAROLL algorithm can work with well-known knowledge bases without adaptation and (2) the user can easily integrate new constraints in order to modify the relational schema.

#### Description of the dataset

Since DBPedia encompasses triples created from the Wikipedia database, it is a massive knowledge graph where not all relations are relevant to our problem. That is why we have extracted a small portion of it, only dedicated to films. The knowledge base used is composed of instances of movies selected from DBPedia, completed with other data from the Internet Movie Database, IMDb[2]. The result is an ontology composed of a single class, <http://dbpedia.org/ontology/Film>, from which radiate diverse datatype properties: <dbpedia:ontology/runtime>, <dbpedia:ontology/country>, ... and also three that we have created, for the IMDb score, the genre, and the number of nominations and awards.

[1] https://www.wikipedia.org/
[2] http://www.imdb.com/

*Table 3.2:* **Discretization of the Movie dataset.** Most of attributes have been discretized in order to form equivalent subsets.

| Attribute | C1 | C2 | C3 | C4 |
|---|---|---|---|---|
| Budget (M$) | 1 - 3.5 | 3.5 - 18 | 18 - 5000 | |
| *Size* | 3098 | 3075 | 3051 | |
| Gross (M$) | 0 - 3.5 | 3.5 - 30 | 30 - 3900 | |
| *Size* | 3079 | 3075 | 3070 | |
| Release Year | 1906 - 1988 | 1988 - 2005 | 2005 - 2017 | |
| *Size* | 3207 | 3091 | 2926 | |
| Runtime (min) | 1 - 96 | 96 - 110 | 110 - 356 | |
| *Size* | 3294 | 2968 | 2962 | |
| ImdbScore (/10) | 1.5 - 6.1 | 6.1 - 6.9 | 6.9 - 9.3 | |
| *Size* | 3186 | 3160 | 2878 | |
| Country | America | Europa | Other | |
| *Size* | 6418 | 1987 | 819 | |
| Win | False | True | | |
| *Size* | 5800 | 3424 | | |
| Nomination | False | True | | |
| *Size* | 4789 | 4435 | | |
| Genre | Drama - Romance Biography | Comedy - Romance Drama | Thriller - Drama Crime | Adventure - Action Family |
| *Size* | 3059 | 2942 | 2019 | 1204 |

### Relational Schema Construction

The attributes are the **origin country** $O$ and the **number of won awards** $W$. Using similarity measures, $O$ is described by the datatype property <imdb:hasForOriginCountry> and $W$ is described by the datatype property <imdb:hasForWonAwards>. $O$ is discretized using the different continents, while $W$ is transformed in a binary variable, that takes True if the film has won an award and False otherwise.

The explaining class is then enriched with the **runtime**, the **budget** and the **release year**, since they describe the film before its release. Each variable is discretized in equivalent categories. Moreover, the website IMDb attributes to each movie one or multiple **genres**: to add this information, we use the *k-means* algorithm to learn four clusters of films, each representing a certain combination of genres. The consequence class is also enriched with the **number of nominations**, the **IMDb note** (average notation of the film given by the users of the website) and the **gross**, since they describe how the public reacted to the film after its release. All of these variables are also discretized in categories of equivalent proportions (see Table 3.2).

Some attributes were discarded since they could not be efficiently discretized (such as the different actors); other because they don't help to check $\mathcal{H}_e$ (such as the Wikipedia ID page). Finally, a cleaning was done to remove all films were the country, ether it had win an award, the ImDB score, the release year and the runtime were unknown (i.e. no missing data). We have 81,000 RDF triplets, representing 9,000 films.

#### Results Discussion

Using the relational schema, a PRM and its corresponding essential graph (Figure 3.10 (a)) are learned, in which two inter-classes relation are not oriented (budget and runtime over the gross), but they can be easily determined thanks to the expert knowledge. Plus, the Consequence class relations' study is straight-forward as they are all already oriented and validated by the user. On the other hand, no Explaining class relation is oriented, meaning that all possible models are Markov equivalent.

However, from this first iteration, the user judges the influence of the release year and the budget over the country incoherent: we thus split the Explaining class into two new classes, $E_0$ with the Country variable and $E_1$ with the rest of the explaining variables, while we keep the Consequence class unchanged. Using this newly defined relational schema we define a second iteration resulting in the learning of a new PRM and its associated EG (Figure 3.10 (b)). The essential graph remains unchanged, meaning that our addition of expert knowledge has not modified the independence model of the first learning. However, thanks to the addition of the new expert knowledge, we are closer to the true causal model.

From there, the expert continues to suggest new modifications of the relational schema in order to integrate more potential causality between the attributes. For each of these modifications, a new model was learned. Figure 3.10 (c) shows the final model resulting of the last iteration. It shows a relational schema far more divided than in the first pass of the algorithm. In this case, we can see that some relations change from (a) and (b) in order to respect the new set of expert constraints, such as for example the addition of a direct relation from the gross towards the win variables.

As stated before, we are not experts of the domain, and the conclusion we draw have to be read keeping that in mind. However, given that our experts constraints are true, that there are no missing variables and that the dataset represents the reality, we can now asses that $\mathcal{H}_e$ is verified: the origin country indirectly causes whether a film receives or not an award, mainly through the release year and the budget variables.

### 3.4.3 Control parameters in cheese fabrication

In this last part, we present an application of our algorithm on a real-life application. The data was given by the INRA's TrueFood project, dedicated to investigate the impact of some combinations of thermophile lactic bacteria (i.e. Streptococcus thermophiluss, Lactobacillus helveticus LH with 2 distinct levels and Lactobacillus delbrueckii LD with 2 distinct levels) on the characteristics

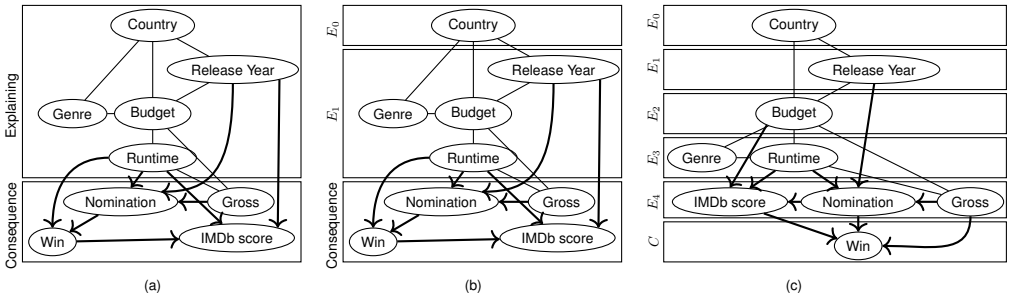



***Figure 3.10:*** **EG and their relational schema learned during the three iterations.** (a) First model learned with only two classes in the relational schema (b) Second model that distinguishes Country from the rest of the explaining variables (c) Third model where multiple classes are created in order to better compartmentalize the variables

of hard cooked cheese. More precisely, they aim to evaluate to what extent this impact can be affected by different factors.

Our dataset focuses on 24 hard cooked cheese of 10kg each. They were manufactured during three weeks in January 2008, and made using 100 liters vats. During the process, they study the effect of the variation of three factors **F1**, **F2** and **F3**:

**F1** = Three different temperatures applied for the milk heating (53°C, 55°C and 57°C). Santiago-López et al. (2018) shows the impact of milk heating and of combination of lactic bacteria during cheese manufacture on the formation of peptides.

**F2** = Two combinations of thermophile lactic bacterias.

**F3** = Three kinds of milks. These milks differ in their protein content and their production conditions. O'Callaghan et al. (2017) shows the impact of the type of milk used for the cheese manufacture (especially the influence of the cows feeding system) on the organoleptic properties of hard cheeses.

Various parameters were also monitored, such as the different measures of proteolysis. In particular, the potentially bioactive peptides content of the cheeses were measured at several steps of the cheese ripening. Finally, the cheeses' sensory properties were also assessed at the end of the ripening step: texture and flavor were evaluated by 11 panelists on a 10 points scale.

For this experiment, we follow the assumption $\mathcal{H}_c$ formulated by the experts: *"**F1**, **F2** and **F3** have a causal influence over the potentially bioactive peptide content of the cheese and its sensory properties"*. They want to asses if F1, F2 and F3 are **control parameters**, i.e. if determining them can influence the result.

| F1. Temperatures | F2. Bacterias | F3. Milk | Number of samples |
|---|---|---|---|
| 53 | 1 | A | 1 |
| | | B | 2 |
| | | C | 1 |
| | 2 | A | 1 |
| | | B | 2 |
| | | C | 1 |
| 55 | 1 | A | 1 |
| | | B | 2 |
| | | C | 1 |
| | 2 | A | 1 |
| | | B | 2 |
| | | C | 1 |
| 57 | 1 | A | 1 |
| | | B | 2 |
| | | C | 1 |
| | 2 | A | 1 |
| | | B | 2 |
| | | C | 1 |

*Table 3.3:* **Cheese plan of experience.** Repartition of the 24 cheeses among the three criteria **F1**, **F2** and **F3**.

### Description of the dataset

The knowledge base describe a transformation process composed of three different steps: Step in the vat, Ripening and Mastication. For each of the 24 studied cheeses, we had:

- **Step in the vat:** is described by three processing control parameters (Temperature, Starters and Type of milk), and two measured (Hardening and Clotting times). The measured times are useful for the evaluation of the **bioactive peptide content**.
- **Ripening:** is described by the measured value of five different concentrations in cheese: butyric acid, propionic acid, acetic acid, free amino acids and free amino groups, which can have an impact on the cheese sensory properties.
- **Mastication:** is described by scores attributed by a panel of 11 judges. For each cheese sample, they evaluate 45 different criteria (e.g. spice aroma, sugar or fat perception), divided in 10 texture attributes and 35 flavor attributes. The scores range from 0 to 10. They are useful to evaluate the **cheese sensory properties**.

  Table 3.3 shows how the main factors were distributed among the cheeses. For each, we also had two measures of time, five concentrations and thirty-nine scores. As a consequence, taking into account the tested control parameters, each of the 24 cheeses was represented by 49 variables.

### Construction of the Relational Schema

From $\mathcal{H}_c$ we deduce that:

- F1, F2 and F3 are directly represented by the three processing control parameters attached to the step in the vat presented in the previous section. They do not need a discretization, as the number of tested modalities is within the range of acceptable for statistical learning

given the length of the database (in our case, 3 to 5 modalities is good).

- The bioactive content is measured by the Hardening and Clotting times, also measured during the step in the vat. The cheese sensory properties are measured by the scores given in the mastication step.

If the explained variables can be easily discretized, we are however presented with a high number of them (especially the notes), which can lessen the quality of the results. As a consequence we first realize a statistical analysis in order to check whether all attributes are useful for the learning or not. This shows notably that the notes attributes during the mastication step have a variance $\sigma < 0.25$. Given the standard variation calculated by $\sqrt{\sigma}$, it means that for these attributes the variation over the whole samples is less than $\pm 0.5$ points. Since the notation is on a scale of 10, we consider it to be too low to observe meaningful variations among the different samples given the discretization. As a consequence, we remove them from the studied set, leaving 39 attributes (9 texture attributes and 30 flavor attributes). Moreover, instead of placing all of the explained attributes in the Explained class, the expert adds a compartmentalization between the times and the notes, since they are not part of the same step: the times are now potentially explaining the notes, as their step is the first to happen.

During the enrichment, we finally add the Concentrations measured in the Ripening step. The final relational schema is presented in Figure 3.11. We can see that the expert have chosen to rename the classes such that each step has a class, that can be subdivided if necessary.

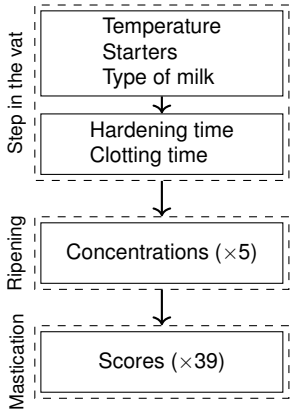



***Figure 3.11:*** **Model constructed from the expert assumption $\mathcal{H}_c$.** Each class representing a step is separated from the ontology's structure validated by the expert. The first Step *Step in the vat* is also subdivided in two classes by expert knowledge.

### Results Discussion

The interest of our approach is the reduction of the space search and the guidance towards a model closer to the reality. As we have seen, we only have 24 repetitions for 49 variables, which

is not ideal considering all the tests and (as we will see) the intersections between some of the variables. The relational schema helps to compartmentalize these different variables with potential causality, allowing to remove models that would not have been interesting (for instance, models that showed that concentrations or scores could explain the temperature).

First of all, we look at the inter and intra classes relation, showed in Figure 3.12. We can see that the three potential control parameters share relations with all the classes, meaning that they directly explain some of their attributes. As a consequence, $\mathcal{H}_c$ is at least partially validated. It is interesting to note that there are a lot of intra-class relations between the different notes variables, meaning that at least some of them are strongly correlated. In an other study, it could be interesting to group those who are correlated in order to reduce the number of variables in the model. Finally, only three sensory notes are not linked at all to any parameter, meaning they are not explained by any variable in the model.

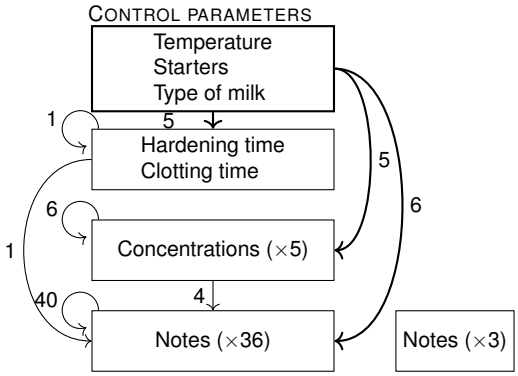



*Figure 3.12:* **Summary of the number of observed inter and intra step relations.** We can see that three notes are not linked to any parameters, and that the others have a lot of intra-step relations, showing that at least some of them are strongly correlated.

While the study of the times during processing and concentrations is pretty straight-forward, all being completely or partially explained by the control parameters, an interesting trend in the sensory notes attributes can be observed while looking at the essential graph. Figure 3.13 presents a part of the learned essential graph, as it mostly focuses on the link between the potential control parameters and the cheese sensory variables. In this figure, the flavor attributes have been grouped in sets in order to enhance the readability. In it, we can see that the six relations shared between the control parameters and the sensory variables are mostly from the type of milk (and one from the starters): those are thus directly explaining some of the variables. More especially, we can see that the explained variables are texture attributes: as a consequence, the type of milk mainly explain the cheese's texture. If we take for instance $T_1$ and $T_2$, we can see that $T_1$ is directly caused by the milk's type: even if the relation is not oriented in the essential graph, it remains an

inter-classes relation and is causally validated by the expert. Moreover, since we have a relation Milk's type $\rightarrow T_1$, then we can orient $T_1 \rightarrow T_2$, meaning that $T_2$ is indirectly explained by the Milk's type. The two texture attributes not represented in the network are related to time and concentration attributes and not directly linked to the control parameters.

On another hand, the flavor attributes are not analyzed as easily. On the contrary, we can notice that a large group of 21 flavor attributes (over the 30), $F_1$, is $d$-separated from the control parameters by another sensory attribute, meaning that this part is in fact equally independent from the control parameters despite being part of the network. This can be explained by multiple facts: (1) as we have seen before, there are a lot of correlations between the attributes; (2) the notes for these attributes have a low variance; (3) they are numerous, despite the low number of repetitions of the experiments (24 cheeses analyzed).

**Definition 10:** *d-separation*. In a directed acyclic graph, a set $Z$ d-separates the sets $X$ and $Y$ if one of these condition hold:

(i) There is a path from $X$ to $Y$ that traverses $Z$

(ii) There is a path from $X$ to $Y$ such that $X... \leftarrow m \rightarrow ...Y$ and $X$ is in $Z$

(iii) There is a path from $X$ to $Y$ such that $X... \rightarrow m \leftarrow ...Y$ such that m is not in $Z$ and no descendant of $m$ in in $Z$.

**Theorem 3:** *d-separation*. If $X$ and $Y$ are not $d$-separated, then they are connected.

**Theorem 4:** *d-separation and independence*. If $X$ and $Y$ are $d$-separated by $Z$, then they are independent if $Z$ is known.

These observations are validated by the experts: considering the milk differences in terms of production conditions and composition, the influence of the milk on the cheese texture was expected. In addition, flavor attributes are indeed more likely to be correlated with each other. In order to disambiguate their analysis, it would be interesting to group them (in order to remove those that are too much correlated), and add more repetitions. Since nearly all flavor attributes are linked together, it could also be interesting to profile the cheeses with their different flavor values. This way, instead of reasoning with all the numerous flavor attributes, we could directly check the influence of the control parameter on the cheese type.

### 3.5 Discussion

The CAROLL algorithm we presented in this section trades the total autonomy of the ON2PRM algorithm with a better adaptability to diverses knowledge bases. The general relational schema

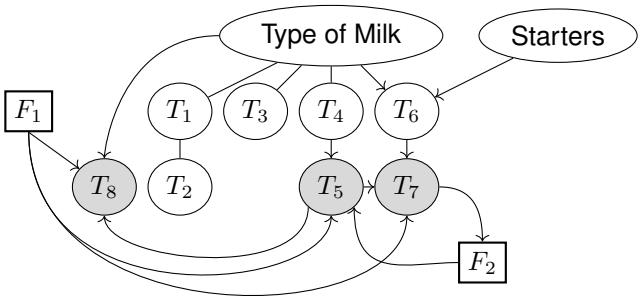



***Figure 3.13:*** **Excerpt of the EG learned for $\mathcal{H}_c$.** We denote $T_i$ the singular texture attributes and $F_i$ the groups of flavor attributes (this grouping is done for a better readability). Grey attributes $d$-separates $F_1$ from the control parameters.

guideline we introduce, the Stack Model, is an helpful tool able to directly model potential causality given a human expert. This input of information helps to guide the learning towards the true causal model. However, this algorithm also present some issues.

- It is very demanding to the expert. As we have seen, the expert has to manually validate the variables, and eventually place them in the relational schema if they have been selected from the enrichment session. If it was an acceptable task for the small examples we have presented, this can become tiresome when dealing with knowledge bases presenting a lot of potential variables.
- It does not directly use the ontology's structure. Since the user has to define all of the relational schema's classes, those are indeed created by a human and never automatically deduced from the object properties as it was the case with the ON2PRM algorithm of Chapter 2.
- It does not handle well the classes instantiated multiple times in a same process.

The last issue is a throwback to one of the problem we raised while presenting the Bayesian networks and probabilistic relational models in Section 1.1.5. In case of an oven instantiated multiple times in the same transformation process, we want to be able to learn only one *"object"* Oven, instead of having to learn multiple times for each instantiations. With the CAROLL algorithm however, this is not possible: each time the oven is instantiated, we have to learn the relation between its attributes.

## 3.6 CONCLUSION

We presented, in this chapter, our algorithm able to semi-automatically build a relational schema from any knowledge base. To do so, we ask to a user expert of the studied domain to formulate a causal assumption $\mathcal{H}$ denoted *"$C_1$, $C_2$,... $C_n$ have a causal influence over $E_1$, $E_2$,... $E_m$"* with $C_i$

potential explaining variables and $E_j$ potential explained attributes. The use of potential causality allows the user to introduce expert knowledge while also respecting the constraints given by the data: if two variables are not dependent, then the expert cannot force the learning of a link between them.

This culminates in an algorithm able to integrate expert's and semantic's constraints in order to learn a model to answer the causal assumption, leading to causal discovery. This causal discovery is however possible only in a very specific setting (dataset complete, that represents the reality, the expert knowledge is true). We propose in this section a protocol to help the experts verify their claim from the assumption they formulated and eventually question the validity of the knowledge base or of their understanding of the domain.

As presented in the Discussion section, this algorithm still lacks of some of the criteria we would have like to find when coupling ontologies and probabilistic relational models. More especially, it would be interesting to study whether integrating more of the way the data is structured while building the relational schema is possible and, if yes, if it can be done while reducing the work required from the expert.

# CHAPTER 4

# SEMI-AUTOMATIC BUILDING OF A RELATIONAL SCHEMA FROM A KNOWLEDGE BASE

**Contribution.**
Munch M., Dibie J., Wuillemin PH. and Manfredotti C. Interactive Causal Discovery in Knowledge Graphs. *In:* PROFILES/SEMEX@ISWC 2019.

In this Chapter we present our last contribution on combining knowledge graphs and probabilistic relational models. We introduce a new algorithm, ACROSS (AutomatiC RelatiOnal Schema conStruction), dedicated to the automatic construction of a relational schema from a knowledge base. Compared to the CAROLL algorithm introduced in Chapter 3, our objective here is to deduce as much information as possible from the knowledge base.

**Section 4.1** gives an overview of the challenges and issues raised by the automatic generation of a relational schema from a knowledge base. First we present the general idea of how to pass from an knowledge base to a relational schema (4.1.1). Then we present the translation rules **R** we have defined (4.1.2). Finally, we present the limit cases and the pre-requisite the knowledge base must have in order to be translated (4.1.3).

**Section 4.2** presents the ACROSS algorithm. It first opens with a comparison between CAROLL and ACROSS's philosophies (4.2.1). Then, we detail its four main parts: the initialization (4.2.2), the automatic (4.2.3) and interactive (4.2.4) parts, and the learning (4.2.5).

**Section 4.3** presents the application of ACROSS on a real knowledge base built from the DBpedia ontology. It presents the context, which is about authors and books (4.3.1); then the applications and experiments (4.3.2), and a discussion on the obtained results (4.3.4).

**Section 4.4** discusses the work presented in this chapter, covering its limits (4.4.1) and highlighting the importance of the expert feedback (4.4.2).

**Section 4.5** concludes this chapter.

## 4.1 Closing the Open-World Assumption

As we have presented in the introduction, ontologies and probabilistic models share different philosophies. While ontologies are dedicated to model expert knowledge to represent a domain, the Bayesian approach favors the statistical analysis of the given data. Both are able to reason within the domain, but they take a very distinct approach when dealing with the data. Indeed, on the contrary of probabilistic models where all cases must be represented, ontologies assume the open-world assumption (OWA), i.e. that they assume that what is not observed is still possible. This is one of the biggest issues we encountered, as moving from ontologies to probabilistic models requires to *close* the OWA: what is not observed is either **impossible** (e.g. there is no particle going faster than the speed of light) or **missing data** (e.g. we have no data for a given experiment because the device supposed to record the measures was not working, or the expert was not interested in it).

In the settings presented in the previous chapters, the expert was drawing himself the limit, by selecting the relevant attributes and defining possible causal explanations; in this section, we will define guideline rules aiming to automatically close the open-world assumption. These rules are thought to be as generic as possible, in order to represent most of the cases.

### 4.1.1 General Idea

The main idea of our automatic transition from a knowledge base to a probabilistic relational model is the same presented in multiple works of the state of the art (Ben Messaoud et al., 2011, Manfredotti et al., 2015, Truong et al., 2005): $O$-classes are directly translated to relational schema's classes. Moreover, to define their attributes, we keep the idea introduced in Chapter 2 and use datatype properties. This raises a first important point: since we need values to learn a model, and that datatype properties are only associated to values when they are instantiated, we **only consider the instantiated part of the knowledge base**. This echoes what we have already defined in Chapter 3: in order to learn the relations between the different attributes, we have to link them, i.e. to find an instantiated chain of object properties between the instances to connect them.

In this section, we will consider the small ontology of Figure 4.1 as an illustration for our examples. It is defined by two $O$-classes $A$ and $B$, and two object properties **OP1** and **OP3** such

that $A \xrightarrow{\text{OP1}} B$ and $A \xrightarrow{\text{OP3}} A$ ($A$ is self-referencing). In this ontology, there are four datatypes properties such that $A$ is the domain of *DP1* and *DP2* and $B$ the domain of *DP3* and *DP4*.

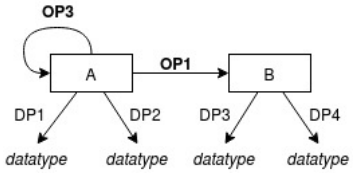



***Figure 4.1:*** **Small ontology's example.** We use this ontology as an illustration for Section 4.1.2. For each rule, we define a set of instances created from this ontology. This allows us to show that even with such a simple example there are multiple and very different possible knowledge bases.

Using this ontology, we can already illustrate the importance of connected instances. If $A$ and $B$ are instantiated, but not the object property **OP1**, then we cannot link the instances of $A$ and $B$, even if they seem connected in the ontology. This is due to the OWA: $A$ and $B$'s instances *can* be connected, but there is no obligation for them to be in order to exist within the ontology[1]. As a consequence, if we want to learn a model as close as possible to the real data, we have to look at the instances, and not at the global schema of the ontology that does not reflect the reality of the instantiated data. Looking at the instances is indeed the only way to (1) gather the values associated to the datatype properties and (2) see how each instance is linked to the others. This last point is used to build the dataset on which is based the relational model learning.

As a consequence, the transformation of a knowledge base to a relational schema is studded with numerous special cases depending on the way each ontology is instantiated. We can deduce that the transition is not easy and requires a good definition. In the next sections, we will present a set of rules **R** to apply in order to deduce the relational schema from a knowledge base.

### 4.1.2 Defining the Transformation Rules

In this section, we will define the rules **R** that describe how to automatically transform a knowledge base into a relational schema. We start by the most generic, **R0**, and continue with others $\mathbf{R}_i$ dedicated to more specific cases.

Each of these rules is illustrated by a small example based on the ontology presented in Figure 4.1. The examples will always be structured in three parts as follow:

(i) An example of the instantiated data in which we assume a common notation for any instantiation: $i_{Xn}$ is the $n^{th}$ instantiation of the $O$-class $X$. This instantiated data and the ontology of Figure 4.1 form the knowledge base used in the example, and represents a particular case where the definition of a new rule is required.

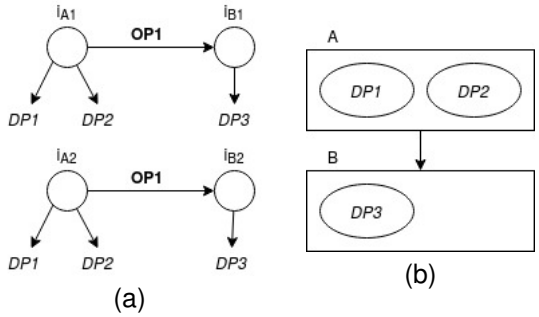
[1] Some ontologies allow to more specify the context in which each instantiation is created (by defining for instance a *min* cardinality). In this thesis, we do not consider such information, as not all ontologies use it.

(ii) The resulting relational schema obtained by applying the rule $\mathbf{R}_i$ to the instantiated data in (i).

(iii) An excerpt of the dataset used to learn the relational model built from the data in (i) and the relational schema in (ii). The goal of (iii) is to show the difficulty of linking the considered instances in order to properly learn a probabilistic model.

#### R0. General Rule

We suppose we have a knowledge base formed by the ontology of Figure 4.1 and the two sets of instances as described in Figure 4.2 (a). When applying **RO**, we create for each instantiated $O$-class its equivalent class in the relational schema of Figure 4.2 (b). Attributes are added using the corresponding datatype properties. As we can see in this example, the datatype property *DP4* is not instantiated in Figure 4.2 (a), and thus is not represented in the relational schema, despite having been defined in the ontology.




**B class dataset**

| Instance | self.DP3 | A.DP1 | A.DP2 |
|---|---|---|---|
| $i_{B1}$ | $i_{B1}.DP3$ | $i_{A1}.DP1$ | $i_{A1}.DP2$ |
| $i_{B2}$ | $i_{B2}.DP3$ | $i_{A2}.DP1$ | $i_{A2}.DP2$ |

(c)

***Figure 4.2:*** **R0. General case.** The classes and the attributes of the relational schema (b) are defined using the instantiated data (a). Since $DP4$ is not instantiated, it is not represented, despite its presence in the ontology of Figure 4.1. The dataset (c) illustrates that we only link the instances that can be linked: there is no object property between the first instantiation of the $O$-class $A$ and the second instantiation of $B$ (i.e. $i_{A1}$ and $i_{B2}$), so they are not associated.

As we can see in Figure 4.2 (c), the attributes values for every relational schema's classes are brought by the instances that have helped to define this class. As a consequence, for each of these classes, a group of the knowledge base's instances are associated. Vice versa, each knowledge base's instance is either associated to a unique class, or it is not part of the final relational schema at all.

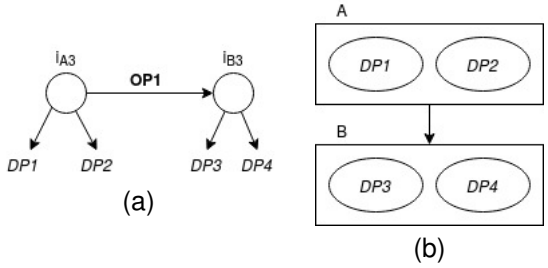
**R0.**

Considering a knowledge base, the relational schema is generated such that:

- The instantiated $O$-classes are transposed as relational schema's classes.
- The instantiated object properties are transposed as relational slots.
- The instantiated datatype properties are transposed as attributes.

### R1. Missing Datatype Properties

For this example, we consider a knowledge base composed of the ontology in Figure 4.1 and the combined datasets of Figure 4.2 (a) and 4.3 (a). In this example we have $i_{B3}$, an instantiation of the $O$-class $B$ with the datatype property *DP4*, which was not the case for the example used to illustrate of **R0**. Considering the new knowledge base, the attribute *DP4* is now also added to the relational schema, resulting in two missing values for $i_{B1}$ and $i_{B2}$ in the **B** class's dataset of 4.3 (c).




**B class dataset**

| Instance | self.DP3 | self.DP4 | A.DP1 | A.DP2 |
|---|---|---|---|---|
| $i_{B1}$ | $i_{B1}.DP3$ | $\emptyset$ | $i_{A1}.DP1$ | $i_{A1}.DP2$ |
| $i_{B2}$ | $i_{B2}.DP3$ | $\emptyset$ | $i_{A2}.DP1$ | $i_{A2}.DP2$ |
| $i_{B3}$ | $i_{B3}.DP3$ | $i_{B3}.DP4$ | $i_{A2}.DP1$ | $i_{A2}.DP2$ |

(c)

*Figure 4.3:* **R1. Missing value.** This example also uses the instantiated set of Figure 4.2. In this case, the datatype property *DP4* is instantiated for some instances (but not all). As a consequence, it is added to the relational schema (b), but it introduces missing values ($\emptyset$) in the dataset (c).

**R1.**

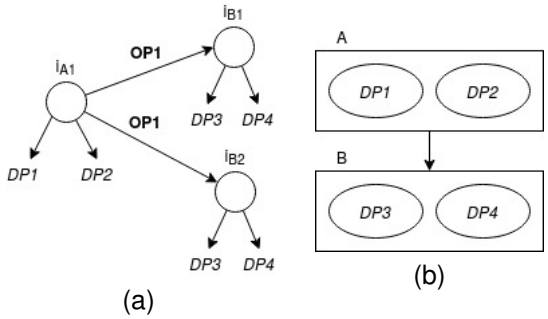
Given a set of instances $S$ of the same $O$-class, and a datatype property $DP$ such that:

- The subset $S_+$ of $S$ is instantiated with $DP$
- The subset $S_-$ of $S$ is not instantiated with $DP$

$DP$ is added to the relational schema as an attribute, to which we associate a missing value for every instance of $S_-$.

### R2. Multiple Instantiations of Object Properties: handling the *domain* classes

Ontologies usually put no restrictions on the number of object properties per instance. If we consider the set described in Figure 4.4 (a), $A$ is instantiated with two **OP1** object properties, each with a different instantiation of $B$. Even if this case is slightly different from the previous one, we however define the same relational schema as before. This is due to the fact that the instances of $B$ do not interact with each other ($i_{B1}$ and $i_{B2}$ are parallel), and thus they can be grouped in the same class.




**B class dataset**

| Instance | self.DP3 | self.DP4 | A.DP1 | A.DP2 |
|---|---|---|---|---|
| $i_{B1}$ | $i_{B1}.DP3$ | $i_{B1}.DP4$ | $i_{A1}.DP1$ | $i_{A1}.DP2$ |
| $i_{B2}$ | $i_{B2}.DP3$ | $i_{B2}.DP4$ | $i_{A1}.DP1$ | $i_{A1}.DP2$ |

(c)

***Figure 4.4:*** **R2. Multiple Instantiations of Object Properties: *domain*.** In this case, we consider $i_{B1}$ and $i_{B2}$ as two parallel instances that do not directly interact, even if they are linked to the same instance $i_{A1}$. As a consequence, the relational schema does not vary from the one defined in Figure 4.3 (b), and the dataset (c) can still be filled.

**R2.**

Given two $O$-classes $A$ and $B$, and an object property $OP$ such that $A \xrightarrow{\textbf{OP}} B$. If there are multiple instances of $B$ involved with a same instance of $A$ through different instantiations of $OP$, then **R0** can still be applied.

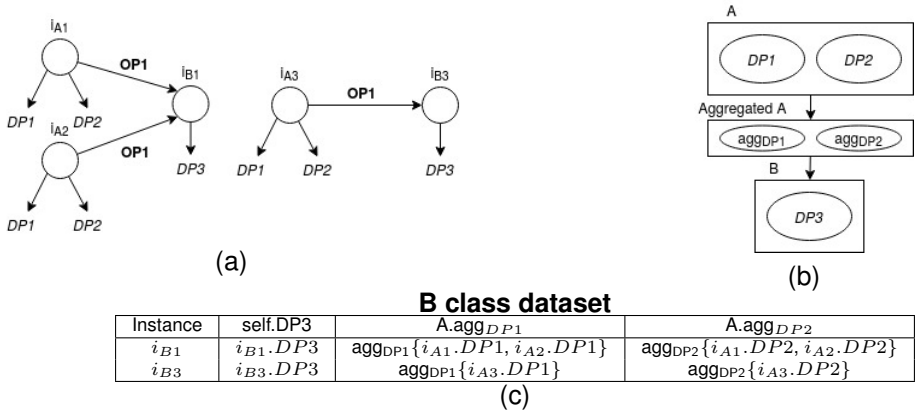
### R3. Multiple Instantiations of Object Properties: handling the *range* classes

The case of multiple instantiations of object properties becomes a bit more complicated when dealing with instances that are range of multiple instances of the same object property. This can be however quite common: multiple students attending the same university ($student_1 \xrightarrow{\textbf{attendsTo}} university_1$, $student_2 \xrightarrow{\textbf{attendsTo}} university_1$), multiple papers written by the same author ($paper_1 \xrightarrow{\textbf{isWrittenBy}} author_1$, $paper_2 \xrightarrow{\textbf{isWrittenBy}} author_1$), etc. In this case, the problem is raised when we want to build the dataset of the *range* class. Indeed, on the contrary of the previous cases where the attributes of linked instances could always be traced (in Figure 4.4 (a) $i_{A1}$ is linked to $i_{B1}$, so in (c) the attribute A.DP1 corresponds to the value $i_{A1}.DP1$), we have here no mean of distinguishing the different attributes. If we apply the relational schema previously defined to the instances proposed in Figure 4.5 (a), which value should be attributed in the class **B** dataset for A.DP1: $i_{A1}.DP1$ or $i_{A2}.DP1$? Since we cannot separate the two values, we need to group them: this is called an **aggregation**.

An aggregation is an operation we can apply anytime when at least one instance is range of several identical object properties. This operation can be a mean, minimum, maximum value, or any possible operation that can be applied to a set of values, such as a boolean (*"Does this set have a value at least superior or equal to 5?"*) or a count (*"How many values are there in this set?"*). It has to be defined by the expert, as it is impossible to automatically deduce the most appropriate

aggregator. If the expert cannot provide one, the link between the two classes in the relational schema is severed, and we cannot make the instances correspond to each other (an example will be shown during the application case in Section 4.3, where the attribute Country's name cannot be aggregated). For the following, given a datatype property $DP$, we denote $\text{agg}_{\text{DP}}$ the aggregated attribute corresponding to $DP$, $\{v_1, v_2, ...v_n\}$ the set of values to which the aggregation is applied, and $\text{agg}_{\text{DP}}\{v_1, v_2, ...v_n\}$ the result of this aggregation.

Figure 4.5 presents a case where an aggregation is needed: for the sake of the example, we consider two sets of instances, one that requires an aggregation ($i_{a1}$ and $i_{a2}$) and one similar to 4.2 (a). The objective is to show that if at least one instance requires an aggregation, we need to aggregate all the attributes.




**B class dataset**

| Instance | self.DP3 | A.agg$_{DP1}$ | A.agg$_{DP2}$ |
|---|---|---|---|
| $i_{B1}$ | $i_{B1}.DP3$ | $\text{agg}_{\text{DP1}}\{i_{A1}.DP1, i_{A2}.DP1\}$ | $\text{agg}_{\text{DP2}}\{i_{A1}.DP2, i_{A2}.DP2\}$ |
| $i_{B3}$ | $i_{B3}.DP3$ | $\text{agg}_{\text{DP1}}\{i_{A3}.DP1\}$ | $\text{agg}_{\text{DP2}}\{i_{A3}.DP2\}$ |

(c)

*Figure 4.5:* **R3. Multiple Instantiation of Object Properties: multiple *range*.** If an instance is the range of several identical object properties(a), then we need an aggregation in order to express the dataset. A new intermediate class, Aggregated A, is also defined in the relational schema (b). It takes the place of the A class in the dataset (c): we do not consider anymore each individual instance from A.

**R3.**

Given two $O$-classes $A$ and $B$, and an object property $OP$ such that $A \xrightarrow{\text{OP}} B$. If there are multiple instances of $A$ involved with a same instance of $B$ through different instantiations of $OP$, then the values of the different attributes attached to $A$ must be aggregated. In the relational schema, a new class *Aggregated A* is created between $A$ and $B$ to which the newly defined aggregated attributes are associated.

#### R4. Distinction Between Different Configurations: The Same Object Property

As of now, we have only studied examples where all $O$-classes instantiations where presenting the same object properties: for each problem, all instantiations of $B$ had, for instance, only one instantiation of **OP1** coming from $A$. This case is usually taken as granted, but due to the OWA,

it is not. In the beginning of this section, we described a case where instances of $A$ and $B$ are not connected despite sharing an object property in the ontology. Similarly, it is possible that in a same ontology some instances of $B$ share an object property with $A$ while others don't. This raises a fundamental problem: is the missing object property $A \xrightarrow{\textbf{OP1}} B$ (1) a fact, or (2) a missing value?

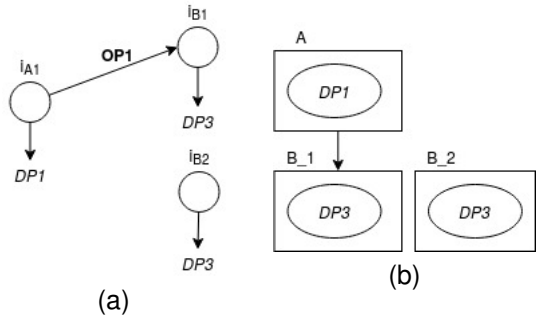
**Example 20.** Suppose an ontology dedicated to describe books, composed of three $O$-classes: $Book$, $Editor$ and $Writer$, and two object properties: $Editor \xrightarrow{\textbf{edited}} Book$ and $Writer \xrightarrow{\textbf{wrote}} Book$. Some instances of books are filled without an editor or a writer, and are missing either the *edited* or *wrote* object property.

(1) In the first case, the omission is voluntary. Some book have no editors (maybe because they have been self-published), and as a consequence it may be interesting to distinguish them from books that have one. The omission is a fact (i.e. *"The book has no editor"*), and not a missing value. The books should therefore be considered as two different classes.

(2) In the second case, however, the missing value indicates that the knowledge base is not complete. Some books have no authors, which seems unlikely: the omission in this case is not voluntary, but an artifact of the dataset. Thus, the books should not be separated on whether they have or not an author, and should be considered the same, in an unique class.

This question, as we introduced before, cannot be answered without expert knowledge, as it directly stems from closing the OWA. However, our goal, in this case, is to offer a first relational schema fully automatically deduced from the knowledge base, that the expert will be allowed to modify. That is why, in our rules, we consider all missing object properties as **a deliberate choice** (first solution (1)), and create as many classes in the relational schema as there are special cases. An illustration is given in Figure 4.6, where the instantiations of the $B$ $O$-class leads to two new $B$ classes in the relational schema.

This choice allows the user to have a good overview of the data. Moreover, it does not make assumptions on the domain and stick as much as possible to the different cases: we suppose that every piece of information, present or missing, has been intentionally designed this way for the knowledge base. This choice also allows the expert to have a brand new look over its data, and eventually discover some patterns (such as books with no authors or publishers) that could have been hidden in the original knowledge base.




**B_1 class dataset**

| Instance | self.DP3 | A.DP1 |
|---|---|---|
| $i_{B1}$ | $i_{B1}.DP3$ | $i_{A1}.DP1$ |
| $i_{B1}$ | $i_{B1}.DP3$ | $i_{A1}.DP1$ |

**B_2 class dataset**

| Instance | self.DP3 |
|---|---|
| $i_{B2}$ | $i_{B2}.DP3$ |

(c)

*Figure 4.6:* **R4. Distinction between different configurations of the same object property.** $i_{B1}$ and $i_{B2}$ both are instances of the $B$ $O$-class, but with different configurations, since one shares an object property with $A$ and the other don't. As a consequence, we distinguish them in the relational schema and create two classes (despite having only one $B$ $O$-class in the knowledge base).

**R4.**

Given one $O$-class $B$ range of an object property $OP$, and $S$ a set of instances of $B$ such that:

- The subset $S_{B^+}$ of $S$ is instantiated as a range of an $OP$'s instance
- The subset $S_{B^-}$ of $S$ is not instantiated as a range of an $OP$'s instance

For each subset, a new class is created in the relational schema. Their attributes correspond to the sets' instances' datatype properties, and the relational slot representing $OP$ is added only for $S_{B^+}$.

### R5. Distinction Between Different Configurations: Different Object Properties

Similarly to the problem we just described, it is also possible that there exist multiple distinct object properties between two $O$-classes. As a consequence, the number of possible combinations is raised.

**Example 21.** Be a knowledge base about a university with two $O$-classes $Teacher$ and $University$ and two object properties $Teacher \xrightarrow{\textbf{hasStudiedIn}} University$ and $Teacher \xrightarrow{\textbf{teachesIn}} University$. There are four possible configurations for the university: (1) universities where there are teachers that teaches and have studied in, (2) universities where there are teachers that only teaches, (3) universities where there are teachers that have only studied in, and (4) universities with no teachers. Note that if the two last cases seem ludicrous, they are, however, allowed by the OWA, once again highlighting the need of a human supervision while dealing with knowledge bases.

Same as the previous problem, we consider, in this case, each combination as a possibility that must be specified: we create as many classes in the relational schema as there are of combinations

among the instantiations, be it from the point of view of the *range* (Figure 4.7) or the *domain* (Figure 4.8).

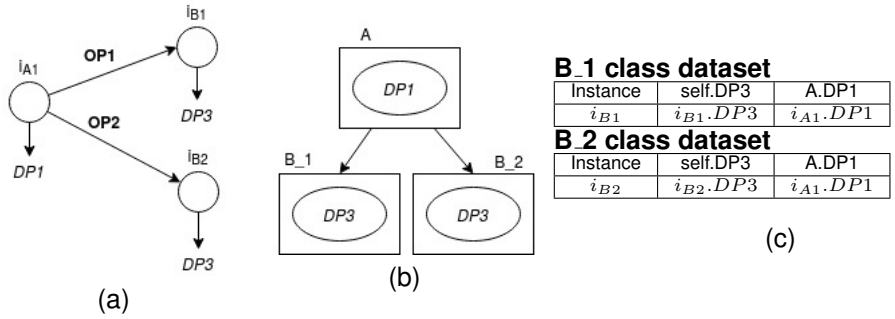



***Figure 4.7:*** **R5. Distinction between different configurations of multiple object properties: *range*.** $i_{B1}$ and $i_{B2}$ are both instances of the $B$ $O$-class, but are linked to $i_{A1}$ with different object properties. As a consequence, they are divided in two different relational schema's classes, B_1 and B_2.

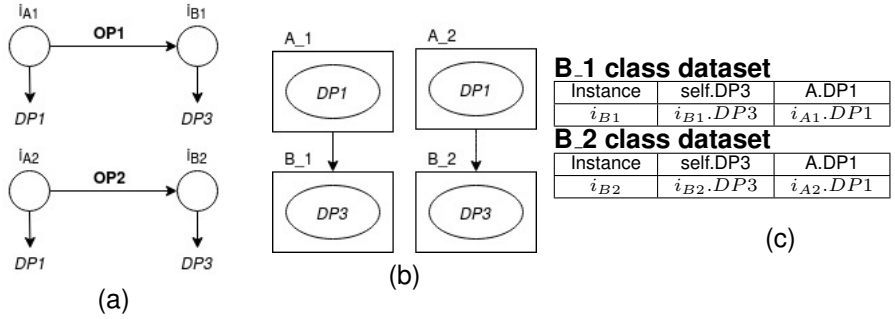



***Figure 4.8:*** **R5. Distinction between different configurations of multiple object properties: *domain*.** $i_{A1}$ and $i_{A2}$ are both instances of the $O$-class $A$, but are the domain of different object properties. As a consequence, they are both treated as different classes in the relational schema.

**R5.**

Given one $O$-class $A$, and the sets $S_{\text{domain}}$ and $S_{\text{range}}$ of object properties that take $A$ respectively for domain and for range. For each instance of $A$ presenting a unique combination of datatype properties from $S_{\text{domain}} \cup S_{\text{range}}$, we create a new class in the relational schema.

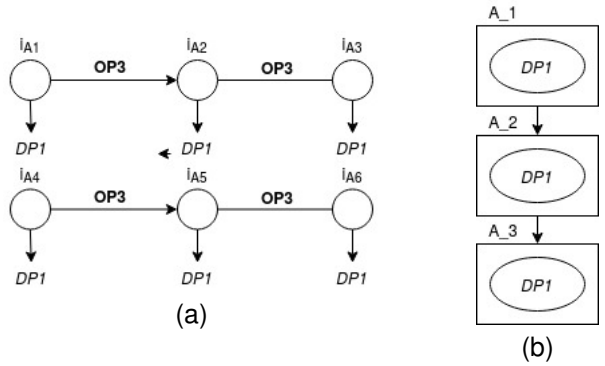
### R6. Self-References in the Knowledge Base

Sometimes, an $O$-class can be self-referencing. If this is possible and common in ontologies, this is however impossible to be directly represented in relational schemas, where cycles are forbidden. On the contrary, if we want to represent a cycle of self-referencing in a probabilistic schema, we need to decompose it into a succession of classes in order to be able to compare them: we compare the first steps of the cycle together, then the second steps, the third... until the last.

**Example 22.** Let's suppose a knowledge base dedicated to represent biological genes. Each gene is composed of a succession of nucleotids, following each other like a string. We have one $O$-class, *Nucleotide*, and one object property, $Nucleotide \xrightarrow{\textbf{isBefore}} Nucleotide$. To describe a gene, we then have a set of $Nucleotide$'s intantiations, each linked to the other through the same *isBefore* object property. To compare two genes, we will then need to compare:

- The first nucleotide of one gene with the first nucleotide of the others
- The second nucleotide of one gene with the second nucleotide of the others
- ...

As a consequence, self-referencing classes are transcribed in the relational schema with respect to their position in the succession of instances. Figure 4.9 gives an example where we added the object property **OP** that allows $A$ to self-reference. In the instantiation example (a) we can see six instances of $A$, which directly translate into three classes in the relational schema.




**A_3 class dataset**

| Instance | self.DP1 | A_2.DP1 | A_2.A_1.DP1 |
|---|---|---|---|
| $i_{A3}$ | $i_{A3}.DP1$ | $i_{A2}.DP1$ | $i_{A1}.DP1$ |
| $i_{A6}$ | $i_{A6}.DP1$ | $i_{A5}.DP1$ | $i_{A4}.DP1$ |

(c)

*Figure 4.9:* **R6. Self-references in the knowledge base.** Each instance is considered as a different class in the relational schema, so it allows a better comparison between the instances: for instance, $i_{A3}$ and $i_{A6}$ are both in the third position, so they are part of the same relational schema's class.

**R6.**

Given one self-referencing $O$-class $A$, we instantiate as many classes in the relational schema as there are instances linked through self-reference.

### 4.1.3 Limits and Conclusion

The rules **R** we have presented share the same objective: guiding the automatic generation of a relational schema by giving general directive to close OWA. They all point towards a same definition of a relational schema's class, which is:

**Definition 11:** *ACROSS's relational schema's class*. A relational schema's class is defined in ACROSS as a group of **instances of the same class**, that are all **domain of the same object properties** and **range of the same object properties**. Its attributes are defined by their instances' datatype properties.

We also give the definition of Parents and Children to describe the relations between the relational schema's classes. However, it is important to note these restrictions are here mainly to help define a first model to ease the expert's load of work, and are not definitive. The modifications they may be subject to will be discussed in the next section.

**Definition 12:** *Parents and Children*. Be $i_X$ an instance of the $O$-class $X$. For every object property in which $i_X$ is involved, we denote

- $i_P$ as a **Parent** of $i_X$ if and only if there is a relational slot such that $i_P \rightarrow i_X$
- $i_C$ as a **Child** of $i_X$ if and only if there is a relational slot such that $i_X \rightarrow i_C$

The way these rules **R** are created also leads to a restriction on the relational slots. As we have seen, to each relational schema's classes are associated to a group of unique instances from the knowledge base, in order to bring the values for the relational classe's attributes. The restrictions of the relational slots assure us the minimum number of missing values.

**Property 3.** Given two relational schema's classes $A$ and $B$ that share a relational slot such that $A \rightarrow B$.

1. For every instance of the knowledge base associated to $B$, it is linked to at least one instance associated to $A$.
2. It exists at least one instance of the knowledge base associated to $A$ that is linked to minimum one instance associated to $B$.

This property guarantees that there are no missing parents: all instances will have at least a parent if a relational slot points towards their class.

This definition allows to clear an issue we encountered in Chapters 2 and 3: a class is now considered as a group of instances that are defined in a same *"context"* (i.e. which are range and domain of the same properties). As a consequence, even if a class is instantiated multiple times (such as an oven in a same recipe), as long as its instances are comparable (share the same context), then they are part of the same relational schema's class.

This means that the generated relational schema owns at the very least as many classes as there are instantiated $O$-classes in the ontology, and potentially much more if there are multiple combinations of object properties (i.e if **R5** is applied). In fact, this leads to one of the limitations of our automatic deduction: while sticking as much as possible to the data described in the

knowledge graph, multiple combinations of object property could lead to an exponential number of relational schema's classes which would potentially bear no sense and be hard to review for the expert. As a consequence, this method of automatic generation is more efficient with classes' instantiations that share a similar pattern: same datatype properties (less missing values) and same object properties (same context, so same relational schema's classes).

However, even if these rules give a good start for the elaboration of the relational schema, they are not able to devise a fully functional one. This is due to our will of always keep the expert close to the decision process. Indeed, in case of aggregations (**R3**), we still need them to define the aggregation function; some datatype properties, even if they are still relevant for the learning (see Definition 9), are not relevant for the domain's representation (for instance, the Wikipedia ID page for the movie example of Section 3.4.2); some distinctions made between classes (**R4** and **R5**) are not true and need, on the contrary, to be combined. Finally, we define the relational slots between the classes according to the ontologies' object properties which are not causal by essence. All of these problems require a human expert verification, that we will define more precisely in the next section.

## 4.2 ACROSS Algorithm

In the previous section, we have presented the main general rules to automatically translate any knowledge base to a relational schema. However, as we have pointed out in Section 4.1.3, these rules alone are not enough and need human expert verification. In this section, we present the ACROSS algorithm and how this introduction of human expert's knowledge, under the form of **user's modifications**, is intertwined with the automatic generation we described.

### 4.2.1 Comparison between CAROLL and ACROSS

The first major difference between CAROLL and ACROSS is the absence of user's assumption. Indeed, since the relational schema is automatically generated from the knowledge base, we don't need a motivation anymore to structure the relational schema. This raises another distinction: on the contrary of CAROLL, where a model was built for a purpose (i.e. to verify the user's assumption), a model built with ACROSS has no stop criteria: it can always be enhanced, and will truly be considered as validated when all edges are causally oriented. This last option is more demanding, because the model is no longer considered as finished when it becomes able to answer the user's assumption. On the contrary, it will be up to the expert to keep in mind what they want to verify with the model and to use it to decide their own criteria of validation. Indeed,

in order to realize these modifications, the expert must have formulated their own questions and hypothesis.

A second difference is the clear separation between the automatic portion of the algorithm (i.e. the first step of the relational schema generation) and the interactive section (i.e. the second step of user's modifications). CAROLL allows a control over every automatic decision, but these several verifications could grow tiresome and repetitive if there are too many variables to check. On another hand, ACROSS allows the user to first have a good overview of the overall data, not only of the classes and object properties, but also on the way they are instantiated (in order to distinguish the particular cases raised by the rules described in the previous section). It is only at this moment, when all the available data is visualized, that (1) the expert can decide whether the data can be fitted in a model (i.e. not too many particular cases, for instance defined by **R5**) and that (2) their contribution is required.

Despite these two main differences, the algorithms are pretty similar. More particularly, once the relational schema has been defined by the user's modifications, the overall learning, verification and causal discovery remain mostly the same. That is why in the following sections we will detail mostly the two first steps, and not the evaluation and causal discovery as it was already covered in Section 3.3.

Figure 4.10 gives an overview of the algorithm and, as in Figure 3.2, shows in dotted lines the steps where expert's inputs are required.

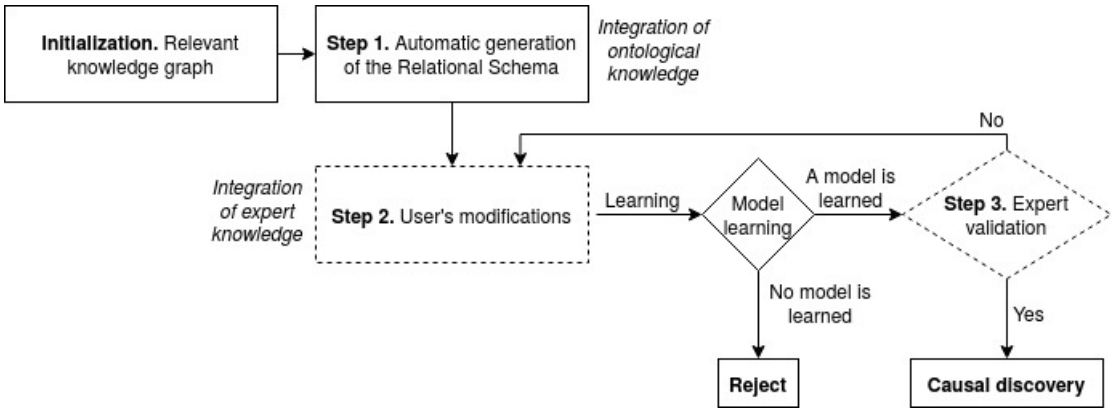



*Figure 4.10:* **Overview of the ACROSS algorithm.** The algorithm is composed of three distinct steps: a fully automatic generation of the relational schema, an injection of expert knowledge and an expert validation. Once these three steps are done, the causal discovery is possible. Each dotted line indicates when a human's input is required.

### 4.2.2 INITIALIZATION

For the initialization, we only require a knowledge base about the domain we wish to represent. As for every previous examples, this knowledge base has to be relevant for causal reasoning, i.e. it has to represent a domain whose variables and causal relations are interesting to model. Moreover, it must respect the same criteria defined in Section 3.2.1: enough classes' instantiations, presence of datatype properties. It has to contain all the important information to build a relational schema. To these criteria, we must also add a new one, that we have sketched out in the discussion of the previous section (4.1.3): the classes must show similar patterns (same datatype or object properties between the instances), and not too many particular cases (such as different object properties combinations).

When a knowledge graph has been selected, a first pass is made to verify whether the object properties create cycles between the classes, as they would prevent the good running of the algorithm. Indeed, as we have presented, the orientation of the relational schema's relational slots is initially made according to the orientation of the object properties. However, this raises a problem when the orientation of the object properties creates a loop between the $O$-classes, as it would be transcribed into a cycle in the relational schema, which would not be possible. Figure 4.11 (a) gives an example of the two kinds of cycles that can be found: a first one across multiple instances, and a second between only two instances. The second one is usually due to symmetric properties, i.e. object properties defined in the ontology as the contraposition of existing properties (e.g. **isMotherOf** and **hasForMother**). Both of these cycles have to be broken in order to use the ACROSS algorithm, the easiest solution being to remove an object property (as shown in (b) which presents one of the possible solution to the problem). Yet, this solution is not easily done, since removing an object property in any cases (and not only when presented with symmetric object properties) also removes semantic meaning and orients the construction of the relational schema towards a certain direction. If the user's modifications described in Section 4.2.4 are able to correct afterwards the relational schema, it would really ease the expert's load of work if the cycle's breaking is already leaning towards a causal orientation. Moreover, if removing an object property can be easily done in the case of symmetric properties, it becomes a bit more uneasy when dealing with cycles spreading over multiple object properties: in this case, the suppression of one (or more) object properties has a deeper meaning, that can bear more consequence in the expression of the knowledge base's semantic.

As a consequence, in case of cycles in the ontology, the best solution for now would be for the expert to preselect beforehand which object properties he would prefer to keep. If not possible, an

easiest (but not as efficient) solution is to break every cycle by removing a random object property in it. As we have seen, if this can be considered when dealing with symmetric properties, this however cannot be properly considered for bigger loops. Section 4.4.2 presents another idea that could guide towards a semi-automatic solution.

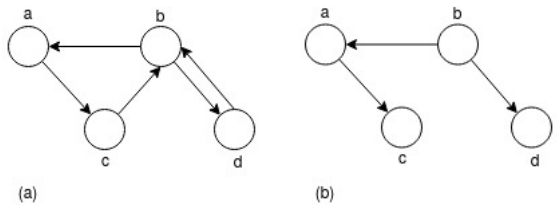



***Figure 4.11:*** **Examples of cycles in an ontology and how to remove them.** (a) This set of instances presents two kind a cycles: (1) between $a$, $b$ and $c$ and (2) between $b$ and $d$. (b) Example of a solution to remove the two cycles shown in (a).

### 4.2.3 Relational Schema's Automatic Generation

When we are assured that the ontology's cycles are dealt with, the automatic generation is done in two steps.

**1. Initialization of the Relational Schema.** The first step of the relational schema's automatic generation is to create the highest classes in the relational schema, i.e. the classes that are not range of any object property. They are defined using the rules we introduced in the previous section. It is important to note that such a class always exist in the ontology, as long as all cycles have been prevented during the initialization. Once defined, they give us a starting point: every other class we build afterwards must be linked to them.

**2. Completion of the Relational Schema.** Once initialized, we look for $O$-classes' instances in the knowledge base such that (1) they are not part yet of a relational schema's class; (2) for every object properties they are involved in, the domain instances belong to one of the relational schema's class (i.e. their parents are already part of the relational schema). For every class defined this way, we complete the relational schema and add them with the corresponding relational slots. We loop this step until there are no more instances to add in the relational schema or that we do not find any more possible child to create new classes.

The relational schema created this way is strongly dependent of the instances represented in the knowledge base, and especially the object properties' orientations. However, as we have already pointed out, this semantic does not always reflect the causality, and the choices we have automatically made are potentially wrong. That is why this relational schema is only a first draft, and can never be considered as valid without the expert's validation first.

### 4.2.4 User modifications

The expert is presented with the created relational schema, to which he is able to bring several modifications which are denoted as **user's modifications**. They are categorized in three different types: **addition**, **removal**, **modification**. Those can be applied at four different granularities in the schema: at the **class**, **relational slot**, **attributes** or **instances** level. The user can technically use as many modifications as they wish, as long as these can be applied, in order to define a relational schema as close as possible to what he knows about the domain.

However, these modifications raise new questions, since they have to be consistent with the knowledge base: for instance, we cannot create a relational slot between two classes that represent $O$-classes which are not linked (directly, or through a combination of object properties) in the knowledge base. As a consequence, their use is limited given the instances represented in the knowledge base. More details about these user's modification and their limits are covered in Appendix A.

### 4.2.5 Learning

Once the relational schema has been approved by the expert, the relational model can be learned. It is done exactly the same way as described in the previous chapters, using the datasets for each class and its potential parents, and taking into account each missing value.

The way the relational schema is generated even allows to solve a recurrent issue mentioned in section 3.5. Indeed, until now, when an $O$-class was instantiated multiple times in different situations (i.e. an oven used for multiple preparation steps in a single recipe), the learning we have described for ON2PRM and CAROLL required to learn as many classes as there were instantiations of the $O$-class (instead of a single one). This issue is solved with ACROSS, where during the generation of the relational schema, we group instances with a same context (i.e. they are instantiated in a same fashion, see Definition 11) in the same classes. This way, all the instantiations of a same $O$-class in the knowledge base, if they are in a same context, can be learned together.

## 4.3 Evaluation

In order to illustrate the ACROSS algorithm, we chose to work with a part of the DBpedia knowledge base dedicated to writers. We want to see whether the information we know about an author and their background has an influence over the books they write.

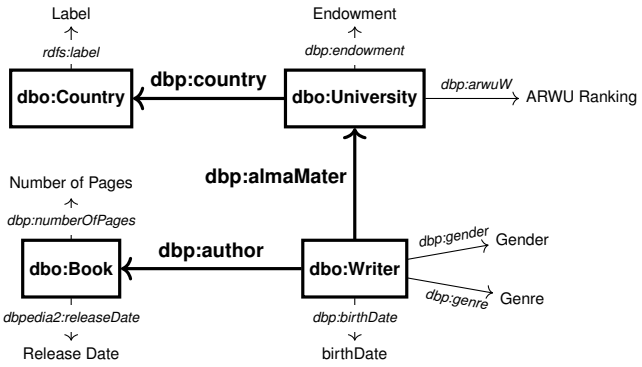



*Figure 4.12:* **Schema of the excerpt of DBPedia used to represent writers.** Only the datatype properties kept in the final relational schema are showed.

### 4.3.1 Domain

The same way we have restricted our domain of study to movies in the previous chapter, we chose here to restrain our study to a much smaller knowledge base[2], dedicated to writers that have written books. During this first pre-selection, we have selected four classes to represent our domain: *Writer*, *University*, *Country* and *Book*. This selected part of the ontology is presented in Fig. 4.12. Considering all possible properties (object and datatype) associated to every instances of these classes, we obtain a dataset of 6,908 triples and 185 writers.

### 4.3.2 Experiments

The chosen ontology can be causally studied: we can formulate causal questions and hypothesis over its different attributes, and have expert knowledge in order to guide the construction of the relational schema. Moreover, it does not present cycles within its classes. As a consequence, the automatic generation can begin without beforehands modifications.

#### Automatic generation

The Writer class is the only class which is not range of a datatype property. As a consequence, it is the first to be added to the new relational schema, alongside datatype properties such as *dbp:Genre*, *dbp:Gender* and *dbp:Birthdate*, which respectively represent the type of books the author writes about, its sex and its birth date. The same way as we were automatically determining whether an attribute is or not relevant for the learning in Chapter 3, we apply a selection over the other datatype properties. For instance, the DP *occupation* is not easily discretizable and, as a consequence, is discarded. The result of this first selection is shown in Figure 4.13 (a).

From the Writer class, the University and Book classes are defined in the relational schema.

[2] https://bit.ly/2X0eeCw

The Book class is pretty straight-forward, and has for attributes the datatype properties *dbp:releaseDate* and *dbp:numberOfPages*, which respectively represent the book's date of release and number of pages. On another hand, the University class presents more difficulties. For example, it introduces an aggregation: indeed, some universities had multiple authors enrolled. As a consequence, during the revision of the relational schema, the expert will need to find an aggregator for the author's attributes. Finally, the university is represented by two attributes: *dbp:Endowment* and *dbp:ARWURanking*, which respectively represent the University's funding and its position in the Academic Ranking of World Universities (ARWU). The result is shown in Figure 4.13 (b).

Finally, the Country class is added, with for attribute *rdfs:label*, which indicates the country's name. Since multiple universities can be registered in a same country, we also need to aggregate the university's attributes. The result is shown in Figure 4.13 (c).

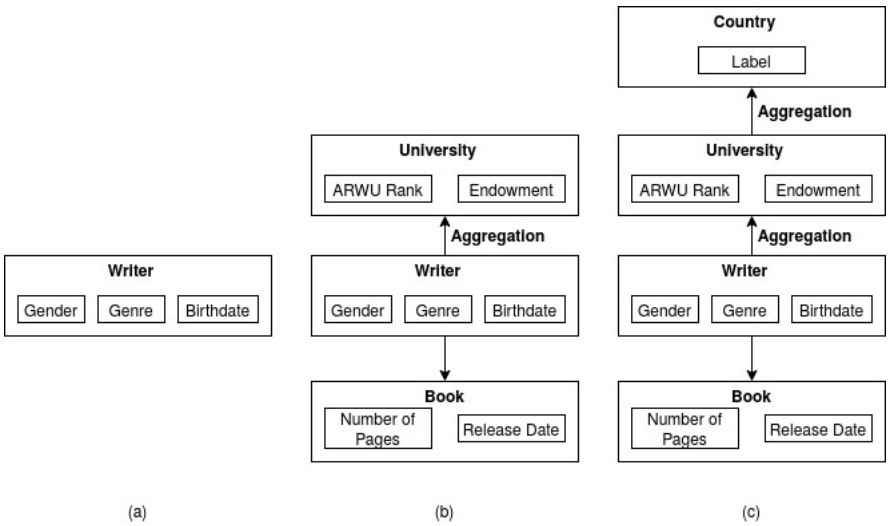



***Figure 4.13:*** **Automatic generation of the author's relational schema.** Each step (a), (b) and (c) corresponds to an automatic addition following the ACROSS algorithm. User's modifications have not been applied yet and an expert's input is still needed for the two aggregations.

### User's modification

According to the original knowledge base, the sequence of relational slots is such that *Country* ← *University* ← *Writer* → *Book* (as shown in Figure 4.13 (c)). However, this presents two anti-causal relations, as it considers that the *Writer*'s attributes might explain the *University*'s which, in turn, might explain the *Country*'s. On the contrary, it seems more logical to build a model such that the *Country*'s attributes might explain the *University*'s which themselves explain the *Writer*'s. As a consequence, the relations between *Country* and *University* on one hand, and the relation between *University* and *Writer* on another hand, are reversed in order to obtain *Country* → *University* → *Writer* → *Book*.

Every time the relational schema is modified, the potential aggregation are re-evaluated. After the changes we introduced, the current aggregation automatically determined is such that:

- The new relation $Country \rightarrow University$ removes the aggregation, since a university can only have one country.
- The new relation $University \rightarrow Writer$ transforms the aggregation between the two classes. Indeed, as it is possible for a university to welcome several writers, it is also possible for some writers to have attended multiple universities. As a consequence, the aggregation remains, but the attributes aggregated change: we now need to aggregate the university's attributes.

Moreover, since the Country is linked to the *Writer* through the University, then in order to link *Country* to *Writer*, we also need an aggregator for the *Country* attributes. Since *Country* represents the country of the university to which the author went, then if the author went to different universities, they probably also went to different countries, hence the aggregation.

The different attributes to aggregate (*University*'s and *Country*'s) are presented to the expert (in this case, us) so they can either propose an aggregation or discard the relational slot.

- ***Writer* and *University*.** For each university, we create two aggregated variables. The **ARWU Rank** is aggregated as the highest university's rank to which the author went; similarly, the **endowment** is aggregated as the highest endowment among all of the universities an author went to.
- ***University* and *Country*.** The only available country's attribute to aggregate is the **label**, for which we, however, found no possible intelligent aggregation. As a consequence, the relational chain between *Country* and *Writer* is severed: during the construction of the *Writer*'s dataset, we do not consider the *Country*'s attribute anymore.

In the end, for each class, we keep the following attributes and discretize them as follows:

- **dbo:Country**: each country is only represented by its **label**. Since the majority of our writers are Anglo-Saxon, we distinguish five categories: USA, Canada, Great Britain, Europe and Asia.
- **dbo:University**: each university is represented by its **ARWU**, and its **endowment**. The endowment is split by its median value. The ARWU ranking is split between the first hundred universities, and the rest.
- **dbo:Writer**: each writer is represented by their **gender**, their **genre** and their **birth date**. Genders are split between male and female, while genres are split between fiction and non-fiction. Birth dates are separated by their median, 1950. Two aggregated attributes have been also added: the **highest rank** among all universities they went to, and the **highest**

***Table 4.1:* Discretization of the *Writers* dataset.** The attributes indicated by a * are aggregated attributes. They refer to the writers.

| Attribute | C1 | C2 | C3 | C4 | C5 |
|---|---|---|---|---|---|
| Number of pages<br>*Size* | $\leqslant 250$<br>292 | $> 250$<br>456 | | | |
| Release Date<br>*Size* | $\leqslant 1980$<br>236 | $> 1980$<br>512 | | | |
| Genre<br>*Size* | Fiction<br>692 | Non Fiction<br>56 | | | |
| Gender<br>*Size* | Male<br>592 | Female<br>156 | | | |
| Birthdate<br>*Size* | $\leqslant 1950$<br>486 | $> 1950$<br>262 | | | |
| Endowment<br>*Size* | $\leqslant 10^9$<br>92 | $> 10^9$<br>57 | Missing<br>22 | | |
| University's Max Endowment*<br>*Size* | $\leqslant 10^9$<br>316 | $> 10^9$<br>419 | Missing<br>13 | | |
| ARWU rank<br>*Size* | $\leqslant 100$<br>46 | $> 100$<br>123 | Missing<br>2 | | |
| University' Max ARWU Rank*<br>*Size* | $\leqslant 100$<br>238 | $> 100$<br>510 | | | |
| Country Name<br>*Size* | U.S<br>125 | England<br>25 | Canada<br>11 | Asia<br>8 | Unknown<br>2 |

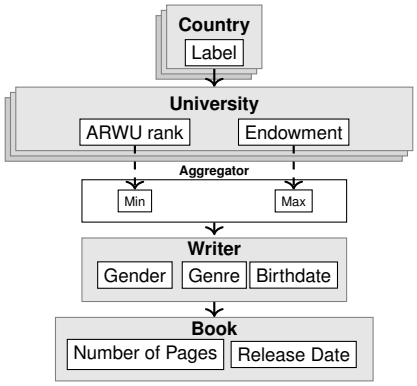



***Figure 4.14:* Relation Schema defined from the knowledge base and the expert's knowledge.** Since a writer can have multiple universities, we introduce an aggregation between the two classes.

**endowment** they went to, with the same discretization used before.

- **dbo:Book**: each book is represented by its **number of pages** and its **release date**. The number of pages is split between books with 250 pages or less and the others; the release date attribute is split between books published before 1980 and those published after.

The final relational schema defined both from the knowledge base and the expert's knowledge is presented in Fig. 4.14.

### 4.3.3 Results

Once the relational schema has been validated, we learn the different classes. Each of them has a different context, so they are all learned separately. Moreover, there is no self-referencing class, so there is no need for memory. Figure 4.15 presents both the probabilistic relational model learned (a), as well as its essential graph (b), which are both analyzed in this section.

**Inter-classes relations.** We have three interclasses relations: one between Label and Endow-

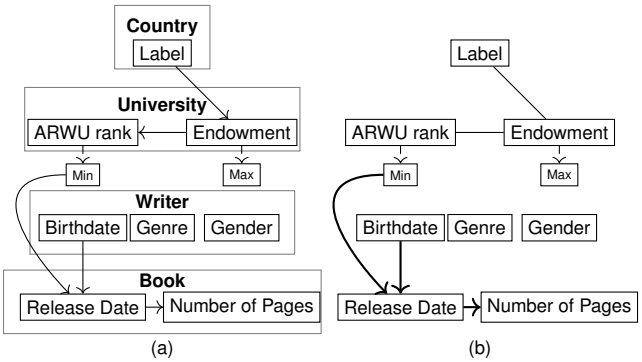



*Figure 4.15:* **Probabilistic relational model learned on a DBPedia extract about authors, and its associated essential graph.** (a) Probabilistic relational model learned. Plain arrows indicates probabilistic relations. (b) Associated Essential graph. Plain arrows indicate essential arcs, non-oriented ones indicate the edges. Dashed arrows only serve as a visual cue to indicate aggregation.

ment, one between the highest ARWU rank and the book's release date, and another one between the author's birth date and the book's release date. Since the relational schema's classes have been built from the knowledge base and the relational slot's direction decided by the user, then we have a causal discovery validated by both the ontological and user's knowledge.

**Intra-class relations.** Among the oriented relations in the essential graph, only one is an intra-class relation, and is thus validated by the data: from Release Date towards Number of Pages. The other intra-class relation (between ARWU Rank and the Endowment) is not oriented, and thus the given ontological and expert's knowledge are not enough to assume the causality between those two attributes.

### 4.3.4 Discussion

For this experiment, we play the role of experts. As such, it falls to us to validate or not the learned model. In this particular case, the relations validated either by expert or ontological's knowledge appears reasonable and agree with common sense. For instance, it seems logical that a university's ARWU rank and its endowment are correlated, and that the endowment itself is explained by the university's country. As a consequence, we can causally accept the relations already oriented in the essential graph or the inter-classes ones.

Only one non-oriented relation remains, between the ARWU rank and the endowment. However, this orientation can be deduced, as we have shown in the previous chapter's Example 19. Since we have *Label—Endowment—ARWU Rank*, but that we know that *Label→Endowment*, thus we can infer that *Endowment→ARWU Rank* (or otherwise we would create a V-structure).

However, as it has already been discussed in 3.3.4, all of this causal discovery remains true

*Table 4.2:* **Joint probability of the attribute releaseDate depending on the attributes writer.birthDate and writer.min_arwu.** The low values 0.01 are an artifact of learning and indicate that these combinations are not encountered in the dataset.

| writer.birthDate | writer.min_arwu | releaseDate | |
|---|---|---|---|
| | | *before_1980* | *after_1980* |
| *before_1950* | *100_or_less* | 0.58 | 0.42 |
| *after_1950* | *100_or_less* | 0.01 | 0.99 |
| *before_1950* | *101_or_more* | 0.44 | 0.56 |
| *after_1950* | *101_or_more* | 0.01 | 0.99 |

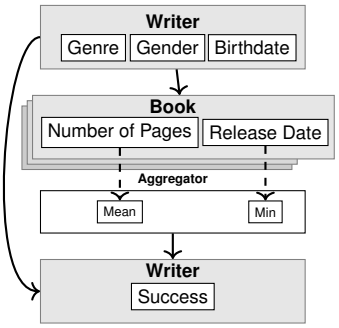



*Figure 4.16:* **Example of a class creation's user's modification.** It is interesting to note that since an author can write multiple books, this creates a new aggregation between the book's attributes and the author's success attribute.

only if we meet some pre-requisite. A further look into our data and the learned relations can indeed question the validity of this model. For instance, it indicates that a book's release date can be explained by both the highest rank of the university its author went to, and this author's birth date (the joint probability is presented in Table. 4.2). It shows that, basically, authors born before 1950 tend to publish slightly more before 1980 when they are from a top-tiers school. On another hand, youngest authors tend to publish after 1980, which at first seems logical: writers born after 1980 would hardly be able to publish books prior to their birth. However, we have no instance in our dataset of books published before 1980 written by persons born after 1950, which explains why we learned this relation. This underlines the importance of a complete and verified knowledge base: if our dataset is representative, then we acknowledge the fact that youngest authors cannot publish before 1980. On another hand, if our dataset is not representative, it means that our learned relation cannot be causal, as we have missing arguments.

Some user's modifications presented in Section 4.2.4 were not used here, such as the class creation. A good example would have been if we had a variable able to indicate an author's success. It would then have been possible to study the impact of an author's books on this newly introduced attribute. To do so, we would have split the Author class in two, to see how an aggregation of the books' attributes would have influenced this variable. Fig.4.16 presents the corresponding relational schema.

## 4.4 Final Remarks

In this chapter, we have presented ACROSS, an algorithm dedicated to learn a probabilistic relational model from an ontology using as much as possible the knowledge of the knowledge base. The most important part of this work is the translation from the knowledge base to a fully functional relational schema. We have tried to present in this chapter all the main possible cases and how to treat them; however, due to the high diversity of the knowledge bases, new cases are still possible. Moreover, even if the rules we have introduced do not cover all possibilities, the main idea behind our work remains the same. Indeed, we aim to automatically create a structure that would allow to **distinguish all particular situations** and to **compare data that are comparable**: if two books are instantiated, one with an author and another without, then they are first considered as two separated entities, even if they are both from the same $O$-class Book.

Defining what is comparable or not is a direct consequence of the difference between knowledge bases and probabilistic models: whereas the first assume that what that has not been defined can still exist (thanks to the Open-World Assumption), the other considers that we must only consider the existing things (meaning in our case that what hasn't been instantiated in the knowledge base is not possible). In order to pass from one model to the other, we thus need to close the open-world assumption. By doing so, we also propose to the expert a novel visualization of their data, by discovering among the instances patterns that might have been hidden in the knowledge base.

The expert is also given the opportunity to modify at will the generated result, by the mean of user's modifications which allow different operations. Similarly as for the rules we defined, we kept in mind during the writing that those modifications have to be the most generic possible and can be used despite the wide variety of knowledge bases. However, some particular cases may have eluded us. Once again, the main idea behind those user's modification is to propose **an intuitive way for the experts to manipulate the relational schema** in order to reflect their own knowledge of the domain.

Once this has been analyzed and the relational schema has been drawn, the learning is similar to what has already been covered in the previous chapters with ON2PRM and CAROLL.

### 4.4.1 Limits

As we have brushed before, the biggest challenge of the ACROSS algorithm is that the automatic decisions the algorithm takes tend to flood the information: each time a particular *context* is discovered (ie. a combination of relational model's classes and object properties), a new class is

added to the relational schema. This might, in the long run, distinguish a lot of possibilities and create too many classes for the expert to manually verify. In this case, the best solution might be to skip the automatic part and ask from the start to the expert to manually design the relational schema. As for every user's interaction design, the most important point is to keep the number of verification's operations lower than the required number of operations for manually doing the task. In our case, if it is simpler for the users to design the relational schema themselves, then ACROSS is not useful. However, it still offers another visualization of the data that the ontology does not automatically provide, and thus can still be useful for the understanding of the domain.

### 4.4.2 EXPERT FEEDBACK

On the contrary of the CAROLL algorithm, we do not include here a mandatory user's assumption. However, as we have seen throughout this chapter, some tasks asked to the expert are tedious and repetitive, such as the cycle breaking or the relational slots orientations. Should we introduce a new type of user's assumption's formulation within the ACROSS algorithm, these two tasks could be more automatized: if we have a cycle between the three $O$-classes $A$, $B$ and $C$, and the user's assumption: *"Do A explain C?"*, then we can easily deduce that the most fitting orientation might be $A \rightarrow B \rightarrow C$, without requiring expert's intervention.

More generally, we tried while designing ACROSS to compartmentalize to the maximum the expert's intervention, in order to keep a clear track of the relational schema, and what has influenced its design: *"Is the creation of this class due to the ontology's semantic, or the expert's user's modification?"* is the kind of question that should be easily answered. Indeed, the bigger a knowledge base is, the more complex the relational schema we have to deal with is. That is why we had at heart to propose a simple solution, easy to understand and to retrace. However, as for CAROLL, this new algorithm also requires the expert to understand and know how the ontology is structured.

## 4.5 CONCLUSION

In conclusion, we have presented in this chapter a third algorithm dedicated to pass from a knowledge base to a probabilistic relational model. The novelty of the algorithm presented in this chapter is the highlight that has been put on the automation of certain tasks. On the contrary of ON2PRM, dedicated to only one particular ontology, or CAROLL, which required a lot of expert's inputs, ACROSS aims to exploit the most knowledge possible from the knowledge graph, leaving the expert to answer the questions that depends only of the domain and that could never

be answered automatically without a very good understanding of the subject by the algorithm (which was the case for ON2PRM and the $PO^2$ ontology).

# CONCLUSION AND PERSPECTIVES

This chapter concludes our work on the topic of combining knowledge bases and probabilistic relational models to outperform both of them. In a first part, we will briefly recall our different achievements and propositions, and confront them to define their strenghts and limits. In a second and final part, we will discuss futur works and possibilities opened by what we have presented.

### SUMMARY OF RESULTS

The introduction presented our problem and to what extent finding a solution to it would help to develop new reasoning tools. Indeed, one of the main issues of automatic learning is the lack of domain's knowledge, which is usually brought by experts. If ontologies greatly contribute by bringing a structuration based on semantics to the data, they however do not allow a strong probabilistic reasoning. On another hand, probabilistic models are great to represent probabilistic relations between different variables in order to describe a domain; they however usually lack expert's knowledge inputs, and domain's semantics is not taken into account during their learning. That is why we chose to focus on the combination of these two different fields, in order to offer a new way of learning more semantically accurate probabilistic models. The interest is double. Firstly, since ontologies are dedicated to represent real-life domains, being closer to their semantics would mean that the learned models would also be closer to the real-life models. Secondly, a model learned using experts' knowledge might be more easily understood by these experts, and as a consequence can be more easily exploited by them.

We present in Chapter 1 the state of the art of the two main topics of this thesis, with an emphasis on the combination of knowledge bases and probabilistic models. As we have seen, this idea has already been brushed by different works; however, many of them assumed chara-

teristics that were not always encountered in ontologies (such as the causal object properties). Moreveover, only a few works exploited the definitions of the ontologies' classes and properties to ease the learning, for instance by using probabilistic relational models (instead of Bayesian networks). In this chapter we also gave a good emphasis on causality, as our main goal is in the end the exploitability of the learned model by the experts: we wish to provide a model able to answer questions about a domain, and explain how the different variables interact with each other.

Chapter 2 presents our first algorithm, ON2PRM (ONtology TO Probabilistic Relational Models). The main idea is to provide an algorithm able to learn a probabilistic relational model from any knowledge base using a specific ontology. In our case, we chose the $PO^2$ (Process and Observation Ontology) ontology, dedicated to transformation processes. The specifity of ON2PRM stems from the fact that the probabilistic relational model's relational schema, that guides the learning, is designed beforehand by a human so that it fits the $PO^2$ semantic. As a consequence, it requires an important upstream work of understanding the ontology's semantic to build the relational schema; but when it is finished, the rest of the learning can be done without requiring anymore expert's intervention (modulo the data discretization and the eventual attributes selection). This chapter essentially focuses on the learning of a probabilistic relational model from a knowledge base and the assessment of the contribution of such a pairing. As we demonstrate, the addition of semantic knowledge helps to guide the learning even with small datasets (i.e. with only a handful of values) toward a better result, closer to the *ground truth*. Our first result is thus a **validation of the interest of coupling knowledge bases and probabilistic relational models**.

***Table 4.3:*** **Summary of the ON2PRM algorithm.**

| Criteria | Algorithm |
|---|---|
| Autonomy | No human intervention required |
| Which knowledge bases | Any knowledge base using the $PO^2$ ontology |
| Principle | (1) A general relational schema have been defined beforehand by an expert using the $PO^2$ ontology. (2) A specific relational schema is built from the knowledge base and the general relational schema. (3) A relational model is learned from the knowledge base and the specific relational schema. (4) A probabilistic relational model is built from the relational model. |

**Contribution.**

Munch M., Wuillemin PH., Manfredotti C., Dibie J. and Dervaux S. Learning Probabilistic Relational Models using an Ontology of Transformation Processes. In: ODBASE 2017, Rhodes.

We discuss in Chapter 3 the limits of ON2PRM, which strongly depends on the relational schema built using PO$^2$. As a consequence, we introduce a second contribution, the CAROLL (Causal Assumption to pRobabilistic RelatiOnaL modeL) algorithm. The novelty compared to the ON2PRM algorithm can be summarized in two points: firstly, it allows the users to define a relational schema on the fly, guided by a causal assumption they have formulated in the beginning of the process; and secondly, it includes a stronger part on the exploitability of the learned model by introducing causal verification and reasoning. The first point allows the transformation of any knowledge base into a probabilistic relational model, as long as the experts can formulate a causal assumption they wish to verify with this knowledge base. On another hand, CAROLL requires more expert's inputs than ON2PRM, as the expert's contribution is higher: for each potential attribute in the relational schema, the expert has to place it, and define the relational schema's classes that will encompass them. The price of this flexibility is therefore the workload required from the expert, which can become overwhelming when the knowledge base is too massive (i.e. too many classes and datatype properties).

This chapter presents two results:

- A **flexible algorithm able to guide an expert to learn a probabilistic relational model from any knowledge base**
- A **causal verification method able to exploit expert understanding of a domain in order to discover new causal knowledge**

*Table 4.4:* **Summary of the CAROLL algorithm.**

| Criteria | Algorithm |
|---|---|
| Autonomy? | Human evaluation required for (1) the construction of the relational schema and (2) the causal verification |
| Which knowledge bases? | Any knowledge base from which a causal assumption can be formulated |
| Principle? | (1) An expert formulates a causal assumption. (2) From this causal assumption a relational model is built by the expert using the knowledge base. (3) The probabilistic relational model is learned from the relational model. (4) A causal verification is used to check wheither the expert validates the model or not. (5) Depending of the validation the causal assumption is either answered or rejected. |

**Contributions.**

- Munch M., Wuillemin PH., Dibie J., Manfredotti C., Allard T., Buchin S. and Guichard E. Identifying control Parameters in Cheese Fabrication Process Using Precedence Con-

straints. *In:* DS 2018, Chypre.

- Munch M., Dibie J., Wuillemin PH. and Manfredotti C. Towards Interactive Causal Relation Discovery Driven by an Ontology. *In:* FLAIRS 2019, Floride, USA.

Chapter 4 introduces our last contribution, the ACROSS (AutomatiC RelatiOnal Schema conStruction) algorithm. Compared with CAROLL, whose relational schema's construction required a user's assumption, ACROSS proposes to automatically generate a relational schema from the knowledge base the user wishes to study. The user is then given the possibility, throught user's modifications, to modify at will this first draft in order to integrate their own knowledge of the domain. The main advantage of this new approach is that we try to exploit more the instances of the knowledge base, and the way these are connected together. This allows the expert to have a new vizualisation of the data to the expert, by highlighting patterns (such as missing values, use of different object properties, ...) that could have been hidden in the knowledge base. If the knowledge base is simple enough, this can eventually simplify the expert's workload by generating a usable relational schema.

In this chapter, we especially put the emphasis on the definition of clear rules that can be applied to any knowledge bases, despite their great variety. However, when this variability becomes too important (especially in case of huge knowledge bases with a lot of instances), the relational schema automatically generated can become a hinder for the expert. That is why ACROSS has to be considered more as a complement of CAROLL instead of a replacement: both of these algorithms work towards the same goal (the learning of a semantical probabilistic model), but put the emphasis on different methods. While CAROLL favors the interactivity with the expert, ACROSS proposes a better compartimentation between automatization and expert's inputs, and tries to go further with the automatic exploitation of the ontology's semantic. Both, in the end, offer a causal analysis of the learned model following the method introduced in Chapter 3.

This chapter brings one results: an **algorithm able to pass from any knowledge base to a probabilistic model while being as automatized as possible**.

**Contribution.**

Munch M., Dibie J., Wuillemin PH. and Manfredotti C. Interactive Causal Discovery in Knowledge Graphs. *In:* PROFILES/SEMEX@ISWC 2019.

*Table 4.5:* **Summary of the ACROSS algorithm.**

| Criteria | Algorithm |
|---|---|
| Autonomy? | A first relational schema draft is fully automatically generated. Human's intervention is required (1) to eventually modify it and (2) to causally validate it. |
| Which knowledge bases? | Any knowledge base whose domain is interesting to study from a causal viewpoint |
| Principle? | (1) A relational schema draft is generated from the knowledge base's instances. (2) Throught user's modifications the expert is given the possibility of modify it. (3) The probabilistic relational model is learned from the relational model. (4) A causal verification is used to check whether the expert validates the model or not. |

#### DISCUSSION AND FUTURE WORKS

If the interest of ON2PRM mostly stems from its hability to demonstrate the contribution of coupling knowledge bases and probabilistic relational models, the two others algorithms, CAROLL and ACROSS, propose different approaches in order to effectively tackle this issue. Depending of the knowledge base we wish to reason with, one of them or a combination of both can be most adapted:

- **A small knowledge base:** if there are only a handful of classes, and not too many particular cases (such as combinations of different object properties for a same class), ACROSS is the most adapted as it can quickly generate a first relational schema. However, if the expert has a very precise idea of what they want to represent, CAROLL can also be applied.
- **An important knowledge base:** if the number of classes and properties is more important, it then depends on the complexity of the instanciations. If there are not too many particular cases, ACROSS would yield the best results while saving time for the expert. On the contrary, if the number of particular cases is too high and creates too many relational schema's classes for the experts to efficiently handle, then they should prefer CAROLL.

In any case, ACROSS still offers a good overview on how the data is instantiated, and might still yield interesting results even if it is not directly used as a relational schema afterwards. Especially, it can give the expert some ideas on how the data is really organised and guide them when designing the relational schema using CAROLL. Moreover, ON2PRM shows that once a relational schema has been designed for a specific ontology, any knowledge base using this ontology can be easily transformed into a probabilistic relational model. As a consequence, once the work of design has been realised, it can then be used for many datasets as long as they lie on the ontology's semantic. This opens a lot of future possibilities for the different domains we have

covered throughout our work.

Firstly, as we have pointed out, the learning of a complete probabilistic and causal model of a domain structured by an ontology can bring a lot to this domain's analysis and the evolution of its knowledge graph. As we have seen in our different experiments, once the model learned, we are able to (1) point out missing entities, such as datatype properties (see the Book example in Example 20) or instances (see the Writers' birthdate problem in Table 4.2) and (2) offer a new overview of the way the instances are linked together with ACROSS. However, probabilistic models also allow to simulate new data through inferences. For instance, once a model about a transformation process is learned, we can infere the different possible results and their probabilities while fixing some parameters (such as the quantity of an ingredient) or, on the contrary, see which parameters are the best to fix in order to obtain a particular result. This may help us complete missing data, and even populate the ontology with new values, once the model has been validated. This is particularly useful for certain domains, such as biology, where data is hard to obtain. Being able to simulate experiments without running them can help the experts to focus on less trivial questions.

Secondly, as we have seen, learning a causal Bayesian network from data alone is a rewarding but complex task. Thanks to the combination with ontologies, we can define semantic constraints that can help the learning by restricting the search space. This allows to work with small datasets from whose it would be, usually, hard to learn even non-causal probablistic relations. More generally, using ontologies for learning probabilistic models can expand their accuracy and representativeness of the target domain.

All of these contributions and possible applications, both for ontologies and probabilistic models, can help experts to better understand their domains by bringing new ways of explaining the different results. Indeed, each of them has to be motivated by arguments from either the expert knowledge, or the statistical learning. As a consequence, the same way ACROSS allowed the expert to check a specific assumption about the domain, each attribute's value could also theorically be explained throught a combination of different arguments from these categories. That is why we would like to focus on the research field of explanation generation. Indeed, as we have seen in the experiment on real data of Section 3.4.3, the learned models can sometimes be overwhelming and difficult to understand for the experts. As we have covered in Section 1, some works have already been done for explaining Bayesian networks (Lacave and Diez, 2002, 2004). However, they mostly focus on highlighting some parts of the networks, without drawing more conclusions. On the contrary, we would like to expand what we have presented throughout this thesis: once a probabilistic model generated from a knowledge base, the main interest would be to define how

to best exploit such a model for the expert's benefit.

To do so, we aim to learn rules from our learned Bayesian network using the knowledge discovered by combining ontologies and probabilistic models. This is similar to rule discovery, an ongoing topic for ontologies that focuses on learning logical rules directly from knowledge bases, Some works have already focused on learning what they define as explanation trees (similar to decision trees) from Bayesian networks (Flores, 2005, Nielsen et al., 2008). The main idea behind the construction of these so-called explanations trees is to focus on the mutual information between the different variables, which represents the quantity of information they share: the higher it is, the closer the variables are. Using this as a measure of "strenght" for the different learned arcs, they chose to keep only the arcs with the stronger values to define the tree, thus highligting the variables that are the closest with each other. Applying this method to our learned model should thus yield interesting results, as it would provide another way of reading the network without having to verify each relation. It would (1) offer a new way of verifying the model and (2) provide to the expert an easy to understand tool for prediction, in complement of the Bayesian network. Moreover, given that the learned probabilistic relational model has been causally validated, it would also allow to build a causal explanation tree, which would allow to tackle causal discovery from a new perspective, which would thus be another possible application of our work. Finally, even if this generation of argument is not possible due to the absence of model validation (by lack of data or causal information), these rules offer new ways for improving the model. If, for instance, we find a quasi-certain rule linking two entities that should not be linked, then it gives the expert new ways of improving the model (for instance by providing new data or revising the relational schema). Therefore, be it with or without enough knowledge (ontological and expert), we would always be able to guide the experts in order to improve their representation of the domain.

In a global context of black-box artificial intelligences, being able to provide explanations to back the system's predictions is a real strenght that these future works will try to uncover, thanks to the combination of knowledge representation and probabilistic learning.

# APPENDIX A

# USER'S MODIFICATIONS

Section 4.2.4 introduces the concept of user's modifications, which are methods given to the user so they can modify the relational schema automatiquely generated in the first part of the ACROSS algorithm. In this appendix, we present in more details these modifications, and to what extent they can be used.

In order to classify them, we define three types of actions (Add, Remove, Modify), and four levels of granularity (Class, Relational slot, Instances and Attributes). Not all combinations are however possible (for instance, "Adding an instance" is not something that can be reasonably expected); on another hand, some combinations require multiple definitions, depending of the context (such as "Adding a class"). Table A.1 summarizes what we will present in the following.

***Table A.1:*** **Every defined user's modification.** All possible and defined user's modifications are presented here.

| Entity \ Action | Add | Remove | Modify |
|---|---|---|---|
| Class | A.0.3 | A.0.1 | |
| | A.0.4 | | |
| | A.0.2 | | |
| Relational Slot | A.0.6 | A.0.7 | A.0.8 |
| Instances | | A.0.9 | |
| Attribute | | A.0.5 | |

### A.0.1 DELETE A CLASS

A class have been created in the relational schema, but it is of no interest for the expert. It can be, for instance, a class with no datatype properties that would not bring any information to the model. As a consequence, the expert is given the possibility of removing it. If this class was

sharing relational slots with other classes, these are removed as well (which could possibly break relational chains).

### A.0.2 FUSE TWO CLASSES OF THE SAME TYPE

As seen before, a same $O$-class in the ontology can lead to two different classes in the probabilistic relational model. For exemple, if we consider the book example (see Example 20), we may have two kinds of books: those whose author is present in the data, and those whose author is not. Following **R5**, this leads by default to the creation of two relational schema's classes, **Book with Author** and **Book without Author**. However, the user is given the opportunity to fuse these two classes: this will aggregate the instances, creating one class **Book** and consider the missing parents (in our example, authors) as missing values.

### A.0.3 DIVIDE A CLASS

Sometimes, attributes of a class can explain and be explained by attributes of another class.

**Example 23.** Suppose a knowledge graph described by two $O$-classes such as $Student \xrightarrow{\textbf{attends}} Course$, and an instanciation set such as Figure A.1 (a). In this case, the relational model automatically deduced (Figure A.1 (b)) does not allow to express possible causality between on one hand, the student's interest and the class's subject and, on another hand the class's difficulty and the student's note.

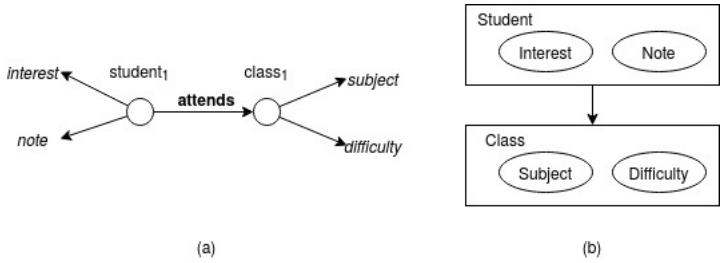



***Figure A.1:*** **Example of a set of instances (a) and its deduced relational schema (b).** In this case, there might be a problem for the determination of the relational slot's orientation: indeed, if we want the student's interest to potentially explain the class's subject, **while also** defining the class's difficulty as potentially explaining the student's note, then we cannot do it with only two classes, regardless of the relational slot's direction.

In this example, we cannot only draw a relational slot $Student \rightarrow Class$ or $Class \rightarrow Student$, since we need arrows to be oriented in both directions. Thus, the expert is given the opportunity to divide a class in order to better sort the relational slots between them. In this specific example, we can create two Student classes: $Student_1$, that contains the student's interest, and $Student_2$,

that contains the note. We then define relational slots such that: $Student_1 \rightarrow Class \rightarrow Student_2$. This result is illustrated in Figure A.2.

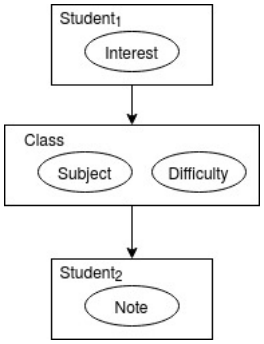



***Figure A.2:*** **Proposition of a division of the Student class.** This new relational schema has been built from the one proposed in Figure A.1 (b). It now allows to express more complex interaction between the different attributes, without creating a cycle in the final result.

More generally, this user's modification allows the expert to divide a class in order to sort the different attributes in multiple class. This allows to better express complex relations of potential causality.

### A.0.4 Create a Mutually Explaining class

In the previous user's modification, we assumed the expert knew how to divide the class (creating $Student_1$ and $Student_2$ and knowing which attributesto put in it). As a consequence, the division is done manually.

In the case where an expert knows that attributes from different classes might mutually explain each other, but without knowing exactly which one, they can create a mutually explaining class. This new class allows to group two or more classes of different types in the ontology to share a relation without it being oriented. During the learning, the attributes' relations are learned without taking care of the original property orientation between the $O$-classes. Then, once they are learned, new classes are created according to (i) the relations' orientation and (ii) the attributes' class. An illustrated example is given in Figure A.3.

The interest of this user's modification is the same as the previous one: allowing the user to better express subtle potential explanations between the different classes' attribute. The main difference with the Divide a class user's modification is that this time, the expert does not know how to divide the class: with this method we offer to automatically deduce it.

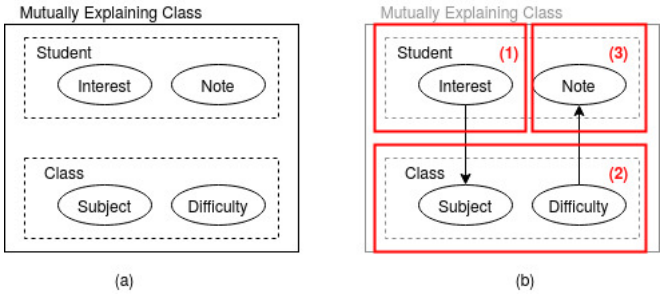



*Figure A.3:* **Example of a Mutually Explaining Class.** The attributes from the Student and the Class are grouped together in a same class (a). After the learning of the relations (b), these classes are divided into different groups depending of (i) the relations' orientation and (ii) the attributes' class. This creates new classes in the PRM ((1), (2) and (3)), which, when defined, allow us to have a relational schema similar to the one of Figure A.2.

### A.0.5 Remove an attribute

On the contrary of the CAROLL algorithm, ACROSS automatically adds all datatype properties to the relational schema as long as they are instantiated datatype properties. This requires a double verification from the expert: (1) whether the attribute is interesting for the problem and (2) whether it is useful for the learning or not (according to Definition 9). If the first verification is completely subjective, the second can be somewhat automatized once the classes are defined and their set of instances known: if values are all constant, or on the contrary all different, then this attribute is not to be kept. A limit can also be drawn in case of too many missing values.

### A.0.6 Create a relational slot

All properties in the knowledge base can sometimes not be enough for expressing the causality between the classes. For instance, we can have a relation such that $A \rightarrow C \leftarrow B$, but in which the expert would like to express the potential causality $A \rightarrow B$. As a consequence, they are given the possibility of defining new relational slots between the relational schema's classes. This user's modification must however respect two conditions: (1) the new relational slot does not create a loop between the classes, and (2) there is a combination of object properties in the knowledge base such that the instances of the two classes we join are linked. Moreover, in case of multiple possible combinations of object properties, the expert must choose which one is represented by the newly added relational slot. Figure A.4 shows an illustration of this problem.

### A.0.7 Remove a relational slot

The same way a class can be removed, a relational slot can be removed in the relational schema. In this case, the learning is not able anymore to find relations between attributes of the two separated

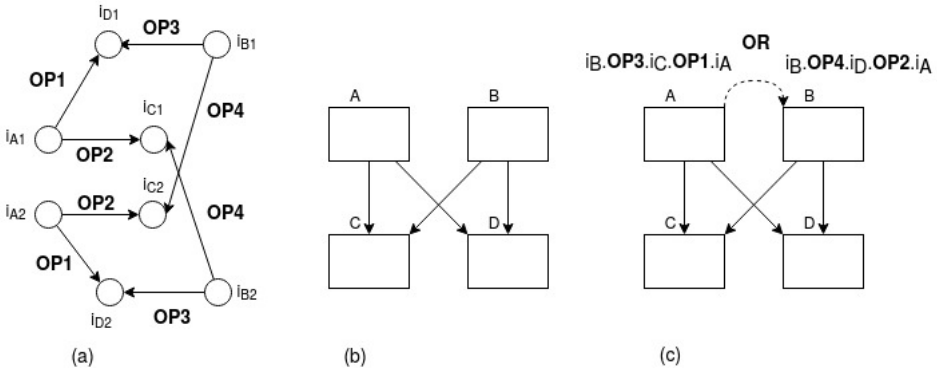



***Figure A.4:*** **Illustration of the creating relational slot user's modification.** Considering the set of instances described in (a), we build the relational schema shown in (b). If the expert wants to add potential causality from the $A$ to the $B$ class, they need to create a relational slot between these classes. (c) presents in dotted line this addition, which is possible since (1) it does not induce a cycle inside the relational schema and (2) the set of instances allows a combination of OP from $A$'s instances to $B$'s instances. However, two combinations are possible: throught **OP1** and **OP3**, or **OP2** and **OP4**, which could possibly lead to totally different ways of pairing the instances: the first way associates $i_{A1}$ with $i_{B1}$, while the second way associates $i_{A1}$ with $i_{B2}$. In these cases, the expert has to choose wich association to favour.

classes as the possible causation is removed.

#### A.0.8 REVERSE A RELATIONAL SLOT

The expert is given the opportunity of reversing the relational slot between two classes, mostly to better express the orientation they think is the most likely causal. The same way as when creating a relational slot, this reversal must insure that no cycle is created in the relational schema.

#### A.0.9 FILTER THE INSTANCES USED FOR THE LEARNING

In some very specific cases, the expert might want to focus on certain instances, and not on the whole set. It is usually the case when they are presented with too many missing values for instances' datatype properties: instead of ruling out the datatype property because it is not useful for the learning, the expert can, on the contrary, choose to remove all instances that are missing this property. This, however, apply a selection filter over the dataset which can eventually deteriorate its representativness.

# APPENDIX B

# RÉSUMÉ EN FRANÇAIS

L'idée de cette thèse est de combiner deux domaines de l'intelligence artificielle, l'ingénierie des connaissances et l'apprentissage probabiliste, afin d'améliorer l'apprentissage de modèles experts et de réaliser de la découverte de savoir.

L'ingénierie des connaissances est centrée sur la représentation de connaissances expertes sous la forme de structures spécialisées, les ontologies. Chaque ontologie est dédiée à la représentation d'un domaine, et le décrit à l'aide de concepts orientés objets, tels que les classes, les propriétés et les instances. Ces structures sont principalement construites manuellement en coopération avec des experts du domaine, et permettent ainsi d'exprimer un savoir particulier, dit expert, qui ne peut généralement pas être représenté de manière automatique dans les apprentissages classiques. Ainsi, l'ingénierie des connaissances permet une représentation poussée des domaines que l'on souhaite étudier, et offre une description poussée des données dont on dispose.

L'apprentissage probabiliste est dédié à l'apprentissage de modèles représentant des domaines à forte variabilité. L'incertitude qui émaille ainsi les données issues de tels domaines (liée par exemple à des mesures manquantes, des modifications intrinsèques, etc.) peut ainsi être représentée de façon précise, grâce à l'utilisation des probabilités. Si l'apprentissage de ces modèles est relativement simple, et peut apporter des résultats satisfaisants même à partir de bases de données de faibles tailles, il est néanmoins possible d'enrichir ce procédé par l'introduction de contraintes supplémentaires. Dans cette thèse, nous présentons donc trois façons d'intégrer les connaissances expertes fournies par une ontologie dans l'apprentissage d'un modèle probabiliste dédiée à la représentation des données décrites dans cette ontologie.

Le **Chapitre 1** se concentre sur l'état de l'art des connaissances sur le domaine et la présentation

des outils utilisés pour la suite. Nous y introduisons la notion de modèles probabilistes, ainsi que les différents modèles maniés tout au long du manuscrit : les réseaux Bayésiens et leurs dérivés orientés objet, les Modèles Relationnel Probabilistes. L'intérêt de la combinaison avec les ontologies et l'utilisation de contraintes durant l'apprentissage est également souligné, à travers des exemples tirés de travaux tirés de l'état de l'art. Dans une deuxième partie, un point sur la causalité et la déduction de connaissances causales est réalisé. Nous y présentons les principales avancées et difficultés, ainsi que les applications possibles dans notre cas (notamment, encore une fois, via l'utilisation d'ontologies).

Le **Chapitre 2** présente nos premiers travaux réalisés dans le cadre de la thèse. A partir d'une ontologie spécifique donnée, PO² (Process and Observation Ontology), dédiée à la représentation de processus de transformation, nous mettons en place un protocole expérimental visant à démontrer l'efficacité de l'introduction de connaissances expertes lors de l'apprentissage d'un modèle relationnel probabiliste. Dans une première partie, nous présentons le domaine d'application ; dans une seconde, nous décrivons notre première proposition, l'algorithme ON2PRM (ONtology to Probabilistic Relational Model). Celui-ci a pour but d'automatiquement s'adapter à un jeu de données décrit par l'ontologie PO² et d'apprendre un modèle probabiliste : s'il permet effectivement une amélioration significative du résultat d'apprentissage, il manque néanmoins d'adaptabilité et ne peut être appliqué à d'autres ontologies. La conception d'une méthode générique est néanmoins compliquée et parsemée de questions diverses.

Le **Chapitre 3** poursuit cette réflexion sur la recherche d'un algorithme plus générique, et propose ici CAROLL (Causal Assumption to pRobabilistic RelatiOnal modeL). L'idée derrière cette proposition a germé avec le constat que les experts désirant étudier leur domaine disposent généralement d'une idée très spécifique de ce qu'ils souhaitent vérifier. Exprimée sous la forme d'une *causal assumption* (affirmation causale) du type "Est-ce que les paramètres A, B, C, ... ont une influence sur les paramètres D, E, F... ?", cette affirmation permet d'exprimer des contraintes favorisant l'initialisation du modèle d'apprentissage. L'efficacité de cette approche est démontrée à travers trois exemples d'évaluation : un premier sur un jeu de données synthétiques, généré pour l'occasion ; un second sur un jeu de données extrait de l'ontologie DBPedia (construite à partir du site internet Wikipedia) ; et un troisième sur une problème concret, en coopération avec des experts de l'INRAE (Institut National de la Recherche pour l'Agriculture, l'Alimentation et Environnement).

Ce chapitre introduit également notre méthode de déduction de causalité à partir d'un modèle probabiliste et des contraintes expertes introduites durant l'apprentissage. Nous y discutons notamment du cadre de leur apprentissage et de leur validation.

Le **Chapitre 4** propose une dernière méthode visant à tirer partie de la structure des ontologies pour favoriser l'apprentissage de modèles probabilistes à partir des ontologies. Il présente ainsi ACROSS (AutomatiC RelatiOnal Schema conStruction), dont l'objectif est la généralisation des méthodes présentées jusqu'à maintenant pour offrir une expérience la plus automatique possible aux experts désirant étudier les données contenues dans une ontologie. Cette méthode soulève néanmoins de nombreux questionnements, abordés durant ce chapitre ; la principale difficulté étant la différence de philosophies entre ontologies et modèles probabilistes. En effet, les ontologies appliquent l'*open-world assumption* (supposition du monde ouvert), qui part du prédicat que tout ce qui n'est pas décrit dans l'ontologie peut exister ; à l'opposé, les modèles probabilistes présupposent que tout ce qui n'est pas présent dans la base d'apprentissage ne peut pas avoir lieu. Cette différence de principe crée une dissonance qu'il est important d'adresser si l'on souhaite apprendre un modèle probabiliste.

Nous concluons cette thèse par une comparaison entre les différentes méthodes, et leurs cadres d'application.

# BIBLIOGRAPHY


R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487—-499, 1994.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

T. Bayes. An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370–418, 1763.

M. Ben Ishak, P. Leray, and N. Ben Amor. Ontology-based generation of object oriented bayesian networks. *CEUR Workshop Proceedings*, 818:9–17, 01 2011.

M. Ben Messaoud, P. Leray, and N. Ben Amor. Semcado: A serendipitous strategy for learning causal bayesian networks using ontologies. *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 182–193, 2011.

P. Besnard, M.-O. Cordier, and Y. Moinard. Arguments using ontological and causal knowledge. *Foundations of Information and Knowledge Systems*, pages 79–96, 2014.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8.

G. Bucci, V. Sandrucci, and E. Vicario. Ontologies and bayesian networks in medical diagnosis. *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 1–8, 01 2011.

B. Buchanan and G. Sutherland. Heuristic dendral: A program for generating explanatory hypotheses in organic chemistry. Technical report, STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE, 1968.

W. Buntine. Theory refinement on bayesian networks. *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52—-60, 1991.

R. N. Carvalho, K. B. Laskey, and P. C. G. Costa. Pr-owl 2.0 – bridging the gap to owl semantics. *Uncertainty Reasoning for the Semantic Web II*, pages 1–18, 2013.

F. Castelletti and G. Consonni. Discovering causal structures in bayesian gaussian directed acyclic graph models. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 2020.

D. M. Chickering. Optimal structure identification with greedy search. *J. Mach. Learn. Res.*, 3: 507—-554, 2003.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of bayesian networks is np-hard. *J. Mach. Learn. Res.*, 5:1287–1330, 2002.

M. Christopher D., R. Prabhakar, and S. Hinrich. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

C. Churchman. *The systems approach*. Delta book. Delacorte Press, 1968.

G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Mach. Learn.*, 9(4):309–347, 1992.

D. R. Cox. *Principles of Statistical Inference*. Cambridge University Press, 2006.

D. Ćutić and G. Gini. Creating causal representations from ontologies and bayesian networks. 2014.

P. C. G. da Costa, K. B. Laskey, and K. J. Laskey. Pr-owl: A bayesian ontology language for the semantic web. *Uncertainty Reasoning for the Semantic Web I*, pages 88–107, 2008.

C. De Campos, Z. Zeng, and Q. Ji. Structure learning of bayesian networks using constraints. *Proceedings of the 26th Annual International Conference on Machine Learning, ICML'09*, pages 113–120, 2009.

C. P. De Campos and Q. Ji. Improving bayesian network parameter learning using constraints. *19th International Conference on Pattern Recognition*, pages 1–4, 2008.

A. Devitt, B. Danev, and K. Matusikova. Constructing bayesian networks automatically using ontologies. *Applied Ontology*, 1, 01 2006.

Z. Ding, Y. Peng, and R. Pan. Bayesowl: Uncertainty modeling in semantic web ontologies. *Soft Computing in Ontologies and Semantic Web*, pages 3–29, 2006.

M. Druzdzel and L. C. Gaag. Building probabilistic networks: Where do the numbers come from? *IEEE Transactions on Knowledge and Data Engineering*, 12:481–486, 2000.

F. Eberhardt. Almost optimal intervention sets for causal discovery. *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 161—-168, 2008.

L. Ehrlinger and W. Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 2016.

E. Feigenbaum and P. McCorduck. *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*. Addison-Wesley Longman Publishing Co., Inc., 1983. ISBN 0201115190.

C. Feilmayr and W. Wöß. An analysis of ontologies and their success factors for application to business. *Data And Knowledge Engineering*, 101:1–23, 2016.

S. Fenz. An ontology-based approach for constructing bayesian networks. *Data & Knowledge Engineering*, 73:73–88, 2012.

M. Flores. Bayesian networks inference: Advanced algorithms for triangulation and partial abduction bayesian networks inference: Advanced algorithms for triangulation and partial abduction. 01 2005.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *International Joint Conference on Artificial Intelligence*, page 1300–1309, 1999.

M. Färber, F. Bartscherer, C. Menne, and A. Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web*, 9:1–53, 2017.

L. Getoor and B. Taskar. *Introduction to statistical relational learning*. The MIT Press, 2007.

C. Glymour, K. Zhang, and P. Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10:524, 2019.

T. R. Gruber. Toward principles of the design of ontologies used for knowledge sharing. *International Journal of Human and Computer Studies*, 43:907–928, 1995.

M. Grüninger and M. Fox. Methodology for the design and evaluation of ontologies. *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995*, 1995.

N. Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. *Information Extraction A Multidisciplinary Approach to an Emerging Information Technology*, 1299:139–170, 1998.

J. Gámez, J. Mateo, and J. Puerta. Learning bayesian networks by hill climbing: Efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22:106–148, 05 2011.

A. Hauser and P. Bühlmann. Two optimal strategies for active learning of causal models from interventional data. *International Journal of Approximate Reasoning*, 55(4):926—-939, 2014.

E. Helsper and L. C. Gaag. Building bayesian networks through ontologies. pages 680–684, 01 2002.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

Z. Huang, J. Yang, F. van Harmelen, and Q. Hu. Constructing disease-centric knowledge graphs: A case study for depression (short version). *Artificial Intelligence in Medicine*, pages 48–52, 2017.

L. Ibanescu, J. Dibie, S. Dervaux, E. Guichard, and J. Raad. Po2- a process and observation ontology in food science. application to dairy gels. *Metadata and Semantics Research*, pages 155–165, 2016.

D. Janssens, G. Wets, T. Brijs, K. Vanhoof, T. Arentze, and H. Timmermans. Integrating Bayesian networks and decision trees in a sequential rule-based transportation model. *European Journal of Operational Research*, 175(1):16–34, 2006.

B. Jeon and I. Ko. Ontology-based semi-automatic construction of bayesian network models for diagnosing diseases in e-health applications. pages 595–602, 2007.

A. Kent, M. M. Berry, F. U. Luehrs, and J. W. Perry. Machine literature searching VIII. Operational criteria for designing information retrieval systems. *American Documentation*, 6(2):93–101, 1955.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193.

S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

C. Lacave and F. J. Diez. A review of explanation methods for bayesian networks. *Knowledge Eng. Review*, 17(2):107–127, 2002.

C. Lacave and F. J. Diez. A review of explanation methods for heuristic expert systems. *Knowledge Eng. Review*, 19(2):133–146, 2004.

J. Li, T. le, L. Liu, J. Liu, Z. Jin, and B. Sun. Mining causal association rules. *Proceedings - IEEE 13th International Conference on Data Mining Workshops, ICDMW 2013*, pages 114–123, 12 2013.

D. Madigan, S. A. Andersson, M. D. Perlman, and C. T. Volinsky. Bayesian model averaging and model selection for markov equivalence classes of acyclic digraphs. *Communications in Statistics–Theory and Methods*, 25(11):2493–2519, 1996.

C. Manfredotti, C. Baudrit, J. Dibie-Barthélemy, and P.-H. Wuillemin. Mapping ontology with probabilistic relational models - an application to transformation processes. pages 171–178, 01 2015.

A. R. Masegosa and S. Moral. An interactive approach for bayesian network learning using domain/expert knowledge. *International Journal of Approximate Reasoning*, 54(8):1168–1181, 2013.

L. Menabrea, C. Babbage, A. Lovelace, and A. L. *Sketch of the Analytical Engine invented by Charles Babbage ... with notes by the translator. Extracted from the 'Scientific Memoirs,' etc. [The translator's notes signed: A.L.L. ie. Augusta Ada King, Countess Lovelace.]*. R. & J. E. Taylor, 1843.

M. B. Messaoud, P. Leray, and N. B. Amor. Integrating ontological knowledge for iterative causal discovery and visualization. 5590:168–179, 2009.

A.-W. Mohammed. Knowledge-oriented semantics modelling towards uncertainty reasoning. *SpringerPlus*, 5, 2016.

M. Munch, P. Wuillemin, C. E. Manfredotti, J. Dibie, and S. Dervaux. Learning probabilistic relational models using an ontology of transformation processes. *On the Move to Meaningful Internet Systems*, 10574:198–215, 2017.

M. Munch, P. Wuillemin, J. Dibie, C. E. Manfredotti, T. Allard, S. Buchin, and E. Guichard. Identifying control parameters in cheese fabrication process using precedence constraints. *Discovery Science*, 11198:421–434, 2018.

M. Munch, J. Dibie, P. Wuillemin, and C. E. Manfredotti. Towards interactive causal relation discovery driven by an ontology. *Florida Artificial Intelligence Research Society Conference*, pages 504–508, 2019a.

M. Munch, J. Dibie-Barthélemy, P. Wuillemin, and C. E. Manfredotti. Interactive causal discovery in knowledge graphs. *SEMEX @ International Semantic Web Conference*, 2465:78–93, 2019b.

R. S. Niculescu, T. M. Mitchell, and R. B. Rao. Bayesian network learning with parameter constraints. *J. Mach. Learn. Res.*, 7:1357—-1383, 2006.

U. H. Nielsen, J.-P. Pellet, and A. Elisseeff. Explanation trees for causal bayesian networks. *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 427—-434, 2008.

T. F. O'Callaghan, D. T. Mannion, D. Hennessy, S. McAuliffe, M. G. O'Sullivan, N. Leeuwendaal, T. P. Beresford, P. Dillon, K. N. Kilcawley, J. J. Sheehan, R. P. Ross, and C. Stanton. Effect of pasture versus indoor feeding systems on quality characteristics, nutritional composition, and sensory and volatile properties of full-fat cheddar cheese. *Journal of Dairy Science*, 100(8): 6053–6073, 2017.

J. M. Ogarrio, P. Spirtes, and J. Ramsey. A hybrid causal search algorithm for latent variable models. *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, 52:368–379, 2016.

P. Parviainen and M. Koivisto. Finding optimal bayesian networks using precedence constraints. *The Journal of Machine Learning Research*, 14:1387–1415, 2013.

J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. 1985.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. ISBN 0934613737.

J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009. ISBN 052189560X.

J. Pearl and D. Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., 2018.

J. Pearl and T. Verma. A theory of inferred causation. *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 441—452, 1991.

J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81—-106, 1986.

H. Reinchenbach. The principle of causality and the possibility of its empirical confirmation. *Hans Reichenbach Selected Writings 1909–1953: Volume Two*, pages 345–371, 1978.

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. page 1135–1144, 2016.

L. Santiago-López, J. E. Aguilar-Toalá, A. Hernández-Mendoza, B. Vallejo-Cordoba, A. M. Liceaga, and A. F. González-Córdova. Invited review: Bioactive compounds produced during cheese ripening and health effects associated with aged cheese consumption. *Journal of Dairy Science*, 101(5):3742–3757, 2018.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

K. Shanmugam, M. Kocaoglu, A. G. Dimakis, and S. Vishwanath. Learning causal graphs with small interventions. *ArXiv*, abs/1511.00041, 2015.

C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948.

S. E. Shimony. Explanation, irrelevance and statistical independence. pages 482—-487, 1991.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT press, 2nd edition, 2000.

S. Staab and R. Studer. *Handbook on Ontologies*. Springer Publishing Company, Incorporated, 2nd edition, 2009. ISBN 3540709991.

P. Suppes. *A Probabilistic Theory of Causality*. Number no 24. 1970.

J. Suzuki. Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the b & b technique. pages 462–470, 01 1996.

L. Torti, P.-H. Wuillemin, and C. Gonzales. Reinforcing the object-oriented aspect of probabilistic relational models. 09 2010.

B. A. Truong, Y. Lee, and S. Y. Lee. A unified context model: Bringing probabilistic models to context ontology. *Embedded and Ubiquitous Computing – EUC 2005 Workshops*, pages 566–575, 2005.

A. M. Turing. Computing machinery and intelligence. *Mind*, 236:433–460, 1950.

W. van Melle. Mycin: a knowledge-based consultation program for infectious disease diagnosis. *International Journal of Man-Machine Studies*, 10:313–322, 1978.

L. Verny, N. Sella, S. Affeldt, P. Singh, and H. Isambert. Learning causal networks with latent variables from multivariate information in genomic data. *PLOS Computational Biology*, 13: e1005662, 2017.

Yi Yang and J. Calmet. Ontobayes: An ontology-driven uncertainty model. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 1:457–463, 2005.

S. Zhang, Y. Sun, Y. Peng, and X. Wang. Bayesowl: A prototype system for uncertainty in semantic web. *Proceedings of the 2009 International Conference on Artificial Intelligence, ICAI 2009*, 2: 678–684, 2009.

Y. Zhang and X. Chen. Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192, 2018.

H.-T. Zheng, B.-Y. Kang, and H.-G. Kim. An ontology-based bayesian network approach for representing uncertainty in clinical practice guidelines. *CEUR Workshop Proceedings*, 327, 01 2007.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Agriculture, alimentation, biologie, environnement, santé (ABIES)

**Titre :** Améliorer le raisonnement dans l'incertain en combinant les modèles relationnels probabilistes et la connaissance experte

**Mots clés :** Modèle relationnel probabiliste, Graphe de connaissance, Découverte causale

**Résumé :** Cette thèse se concentre sur l'intégration des connaissances d'experts pour améliorer le raisonnement dans l'incertitude. Notre objectif est de guider l'apprentissage des relations probabilistes avec les connaissances d'experts pour des domaines décrits par les ontologies.
Pour ce faire, nous proposons de coupler des bases de connaissances (BC) et une extension orientée objet des réseaux bayésiens, les modèles relationnels probabilistes (PRM). Notre objectif est de compléter l'apprentissage statistique par des connaissances expertes afin d'apprendre un modèle aussi proche que possible de la réalité et de l'analyser quantitativement (avec des relations probabilistes) et qualitativement (avec la découverte causale). Nous avons développé trois algorithmes à travers trois approches distinctes, dont les principales différences résident dans leur automatisation et l'intégration (ou non) de la supervision d'experts humains. L'originalité de notre travail est la combinaison de deux philosophies opposées : alors que l'approche bayésienne privilégie l'analyse statistique des données fournies pour raisonner avec, l'approche ontologique est basée sur la modélisation de la connaissance experte pour représenter un domaine. La combinaison de la force des deux permet d'améliorer à la fois le raisonnement dans l'incertitude et la connaissance experte.

**Title :** Improving uncertain reasoning combining probabilistic relational models and expert knowledge

**Keywords :** Probabilistic Relational Model, Knowledge graph, Causal discovery

**Abstract :** This thesis focuses on integrating expert knowledge to enhance reasoning under uncertainty. Our goal is to guide the probabilistic relations' learning with expert knowledge for domains described by ontologies.
To do so we propose to couple knowledge bases (KBs) and an oriented-object extension of Bayesian networks, the probabilistic relational models (PRMs). Our aim is to complement the statistical learning with expert knowledge in order to learn a model as close as possible to the reality and analyze it quantitatively (with probabilistic relations) and qualitatively (with causal discovery). We developped three algorithms throught three distinct approaches, whose main differences lie in their automatisation and the integration (or not) of human expert supervision. The originality of our work is the combination of two broadly opposed philosophies: while the Bayesian approach favors the statistical analysis of the given data in order to reason with it, the ontological approach is based on the modelization of expert knowledge to represent a domain. Combining the strenght of the two allows to improve both the reasoning under uncertainty and the expert knowledge.

**Université Paris-Saclay**
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France