Uncertain Reasoning in Rule-Based Systems Using PRM

Gaspard Ducamp, Philippe Bonnard, Pierre-Henri Wuillemin

The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Uncertain Reasoning in Rule-Based Systems Using PRM

Gaspard Ducamp, Philippe Bonnard
IBM France Lab
9 rue de Verdun, 94250 Gentilly, France
gaspard.ducamp@ibm.com, philippe.bonnard@fr.ibm.com

Pierre-Henri Wuillemin
LIP6 (UMR 7606), Sorbonne Université
4 place Jussieu, 75005 Paris, France
pierre-henri.wuillemin@lip6.fr

Abstract

Widely adopted for more than 20 years in industrial fields, business rules offer the opportunity to non-IT users to define decision-making policies in a simple and intuitive way. When used conjointly with probabilistic graphical models (PGM) their expressiveness increase by introducing the notion of *probabilistic production rules* (PPR). In this paper we will present a new model for PPR and suggest a way to handle the combinatorial explosion due to the number of parents of aggregators in PGM such as Bayesian networks and Probabilistic Relational Models in an industrial context where marginals should be computed rapidly.

1 Context

Business Rules Management Systems (BRMS), such as *IBM Operational Decision Manager* (ODM), are developed since the 90's to facilitate editing, authoring, deploying and executing business policies by domain users, in the form of conditions/actions rules. Syntactically close to the business language, these ease the translation of decision-making and business strategies, making them accessible to users with no programming experience.

Many approaches have been used in the rule-based system community to deal with uncertainty, using *certainty factors* (Buchanan and Shortliffe 1984), *likelihood ratio* (Hart, Duda, and Einaudi 1978) or even *fuzzy logic* (Zadeh 1965). However, there was some limitations using such methods, mainly due to interpretation being incoherent with probability theory (Heckerman et al. 1992) or inconsistency in the conclusions when performing chains of inference (Ng et al. 1990). Bayesian techniques, mostly based of Bayesian networks (Pearl 1988; Weber et al. 2012), have been used to model domains with uncertainty but are not suited for complex systems involving high design and maintenance costs (Koller et al. 1997). Another solution could be to use models that combines first-order logic and probabilistic reasoning, such as Markov logic networks (Richardson and Domingos 2006), but their abstract structure is incompatible with business rules' principles.

This paper will start with a short introduction on probabilistic rules and their relevance in an industrial context as well as their current limitations. We will then illustrate how we propose to inscrease their expressivity using a tight coupling of a BRMS with an object-oriented probabilistic graphical model as well as a way to compute a certain type of probabilistic aggregators efficiently. To illustrate our paper we will take the example of a state willing to monitor and manage its cities' water resources according to their daily consumption and possible episodic drought.

2 Probabilistic production rules

When using a BRMS, users have to define the objects that will be manipulated by the rule engine through classes and attributes declaration. They will be dynamically instantiated in a working memory during the execution of the program. In our case the working memory will contain objects representing cities, water towers and level sensors (that could be either working or broken), as shown in the Figure 1.

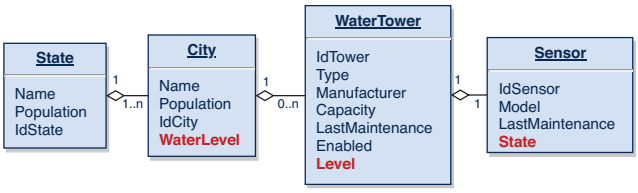

Figure 1: UML class diagram for the water shortage example

Alongside the object data model, users have to define a set of rules. The activation/execution of those rules is managed by inference algorithms such as RETE (Forgy 1982). The rule presented in Figure 2, for example, is used to identify broken sensors inside water towers, allowing technicians to be alerted when a maintenance is required.

However one would like to add uncertainty in rules, for instance to perform predictive maintenances rather than corrective ones, reducing the risk of unexpected breakdowns. Studies including (Agli 2017) showed that a loose coupling between a rule engine and probabilistic graphical models allowed reasoning about uncertain data.

```
1: rule SensorMaintenance {
2:   when {
3:       c: City() ;
4:       wt: WaterTower() in c.watertowers;
5:       s: Sensor(s.state==broken) in wt.sensor;
6:   } then {change the sensor}}
```

Figure 2: Example of rule used in a BRMS

A Bayesian network (BN) is a graphical compact representation of a joint probability distribution over a set of random variables where each node is associated with a conditional probability table (CPT) that contains the conditional probabilities of the random variable with respect to its parents. BNs are inadequate for modeling large scale world; they quickly loses their expressivity due to the large number of obtained variables.

Probabilistic Relational Models (PRM), on the contrary, are combining notions from BNs and from the paradigm of object-oriented languages (Torti, Wuillemin, and Gonzales 2010), where the focus is set on classes of objects and relations between them (using classes, attributes, relations, interface, inheritance and instantiations). Such expressiveness makes it possible to answer the problems of reusability and scalability of graphical models (Medina Oliva et al. 2010). The structuration of information in a PRM being close to the one in the object data model of the rule engine allows us to generate it directly from an annotated version of the model. Figure 3 shows an example of relation schema for PRM classes generated from the model described in Figure 1 where red attributes correspond to probabilistic variables that will have to be handled by a probabilistic engine. A class *City* contains an attribute called *waterlevel* characterizing the availability of water resources: it is acting as a sum aggregator over the *level* attribute of the water towers present in the reference slot (dashed oval). Each tower is linked to a sensor analyzing its water level.

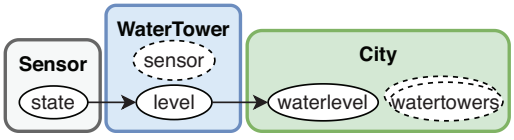

Figure 3: PRM class dependencies schema

By manipulating the object model during compilation, the previous works allowed users to define a first set of probabilistic rules. (Agli 2017) raised two major issues:

- **Business user friendliness**: such rules can be difficult to define and to understand by a business user, expressing a probability on particular conditions requiring a deep level of knowledge of the probabilistic models used.
- **Performance**: ODM provides the ability for users to define different types of conditions (for example, using filters, aggregators, and nested conditions). Neither these constructions, more complex, nor their impacts on the performances have been studied.

3 A new definition of PPR

To address the business user friendliness issue, we have redefined the treatment of uncertainty in the expression of rules by replacing the probability thresholds attached to single variables by a notion of acceptable risk on the evaluation of the conditions of the rule as a whole. The action part of a rule will therefore be executed only if the set of conditions is verified with a probability greater than the defined acceptable risk. This will allow our probabilistic rules to be both more complex and intuitive, but it requires a redefinition of the rules compilation phase to redistribute the overall risk to each individual condition. In the example in Figure 4, a city will restrict its water usages if there is a high probability that either its temperature is high or a low water resources are lower than 10^4 megalitres.

```
1: rule RestrictAccessToWater {
2:   when {
3:       c: City(c.temperature > 40°C or c.waterlevel <10^4) ;
4:   } [with probability > .9] then
5:       {restrict water usage to key functions}}
```

Figure 4: Syntax of a probabilistic rule

To achieve this, it is necessary to generate the PRM from the object model but also from the set of rules. Since the rule engine can not interpret a probability over a set of conditions it is necessary to change its toolchain and intervene during the rewriting phase to make such rules usable.

Compilation, PRM model and rule rewriting

Adapting the PRM model begins with the creation of a new class for each rule (in red in Figure 5), then (i) the predicates present in the conditions are added to the class, as well as the operators that connect them; (ii) arcs are added and operator's CPTs generated according to their nature; (iii) the conditions are finally connected to a boolean random variable called $risk$ whose value will be queried at each inference, it acts as a conjunction between the conditions.

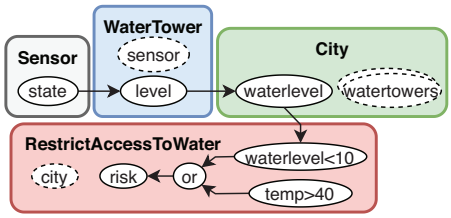

Figure 5: Class dependencies schema of the enhanced PRM

The rule evaluation will be based on the use of the probabilistic engine. Once the PRM enhanced, the probabilistic rules are rewritten; after all the conditions, an evaluation of the comparison between the value of the risk node and the specified threshold is added. It is at this level of the rule that the main interaction between the rule engine and the probabilistic engine will occur. Figure 6 shows how the previous rule is rewritten.

```
1: rule RestrictAccessToWater {
2:   when {
3:       c: City ();
4:       evaluate (PRMengine.getRisk("WaterShortageMode", [c],
                                    [c.temperature>40] ) > 0.9) ;
5:   } then {restrict water usage to key functions}}
```

Figure 6: Syntax of the rewritten rule

When executing the rule and if the rule engine finds elements verifying the conditions in the working memory, the calculation function of risk of the engine is called with the following parameters (i) the name of the class of the rule, in order to instantiate it in the probabilistic engine; (ii) the objects in the condition part that are necessary to evaluate the value of the risk variable; (iii) the value of the deterministic elements encoded in the class, used as evidences.

Runtime, PRM system and working memory

Once the PRM model is defined, the instances are created by mapping the object existing in the working memory into a PRM system. As said before an object corresponding to the rule is instantiated as well during the risk evaluation process. Figure 7 shows such a system in a case where the working memory contains two cities, each linked to a certain number of water towers. If we were computing the risk value given that the city c_1 is being evaluated an object r of type *RestrictAccessToWater* would be created, $r.city$ mapped to c_1 and the value of $r.temp > 40$ updated with the truth value of $c_1.temperature > 40$.

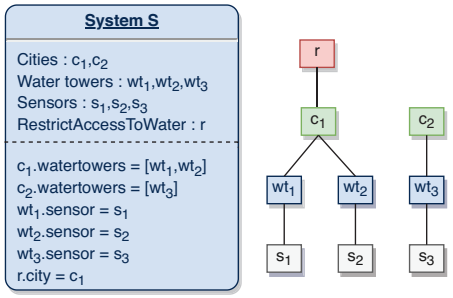

Figure 7: Relational skeleton based on the schema in Fig. 5

From a relation skeleton we can generate a BN called *grounded BN*, as shown on Figure 8; we create a node for each attributes of the objects in the skeleton and linked them according to the dependencies in the PRM model. Once the grounded BN generated, we can compute the marginal of the $r.risk$ node given all the evidences. The complexity of an inference in a BN is NP-Hard (Cooper 1990), growing exponentially in the tree-width of the network, which is bounded below by the size of the largest family (a node and its parents) (Robertson and Seymour 1986). This is one of the major issues when dealing with probabilistic aggregators, especially when they have a high number of parents.

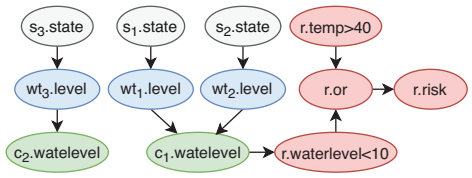

Figure 8: Grounded BN based on the Figure 7

4 Aggregators and combinatorial explosion

Aggregation functions can be considered as a way to summarize information over a set of data. (Jesus, Baquero, and Almeida 2015) present many algorithms about distributed data aggregation, but none specific to the case of probabilistic aggregation in BNs. For probabilistic aggregation we need to provide, for each possible instantiation of the aggregated objects, a value in its CPT. Here, we provide more precise definitions (1 and 3 are given by Jesus, Baquero, and Almeida), and consider that the process consists in the computation of an *aggregation function* defined by:

Definition 1 *(Aggregation function) An aggregation function f takes a multiset of elements from a domain I and produces an output of a domain O, $f : \mathbb{N}^I \mapsto O$*

Definition 2 *(Probabilistic aggregation function) An aggregate function $f : \mathbb{N}^I \to O$ is a probabilistic aggregate function if a function g exists such that:*

$$g : \mathbb{N}^I \times O \mapsto [0,1] \quad \forall x \in \mathbb{N}^I, \sum_{y \in O} g(x,y) = 1$$

In the case of our $waterlevel$ aggregator, if our city is linked to 7 water towers (each charaterized by 11 values), $f : [\![0,10]\!]^7 \to [\![0,70]\!]$ and we will need to store $71 \cdot 11^7$ values. Table 1 shows the inference time using Lazy Propagation algorithm (Madsen and Jensen 1999) and generated file size (from the grounded BN), allowing us to have an idea of the space and time complexity of such cases, even for a small number of parents. We will denote the set of n's parents as $\mathrm{Pa}(n)$ and $|\mathrm{Pa}(n)|$ its size.

Table 1: Inference time to compute city's water level and size of the generated file, depending on $|\mathrm{Pa}|$

$\lvert\mathrm{Pa}\rvert$	time	file size
3	8e-2s	89Ko
4	1.4e-1s	1.2Mo
5	1.9s	16.8Mo
6	30s	241.7Mo
7	> 6h*	2.81Go

Some aggregation functions can be, however, computed by computing the aggregate for subsets, and then aggregating these results, reducing the size of the concerned CPTs.

Definition 3 *(Self-decomposable aggreation function) An aggregation function $f : \mathbb{N}^I \to O$ is said to be self-decomposable if, for some merge operator $\diamond$ and all non-empty multisets X and Y, $f(X \uplus Y) = f(X) \diamond f(Y)$.*

In the above, $\uplus$ denotes the union of multiset. Given that the aggregation result is the same for all possible partitions of a mutliset, it means that the operator $\diamond$ is commutative and associative. Many probabilistic aggregators such as MIN, MAX and SUM are self-decomposable, as shown below :

$$\text{SUM}(\{x\}) = x, \quad \text{SUM}(X \uplus Y) = \text{SUM}(X) + \text{SUM}(Y)$$

To face our issue with aggregation being a bottleneck we will manipulate the grounded BN before the generation of its attributes' CPTs. When the BN contains a self-decomposable aggregator n with $|\,\text{Pa}(n)| > 2$ we can create intermediate aggregators between its parents, grouping them by pairs, then linking these aggregators by pairs themselves. Figure 9 shows how such manipulation is changing a simple BN with one sum aggregator having 5 parents. In this example, each attribute $x_i \in \{x_1, ..., x_n\}$ can take 10 values. In the decomposed counterpart the largest CPT contains 2500 times less parameters than in the original one.

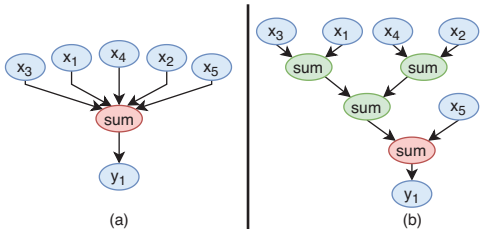

Figure 9: (a) is a BN before its decomposition, its largest CPT contains $50 \cdot 10^5$ values. (b) is the same BN, after decomposition. Its largest CPT contains $50 \cdot 40 \cdot 10$ values.

Table 2 shows the time needed to decompose and compute marginals in the water shortage example. As expected reducing the size of the aggregator's node helped us reducing the complexity of the inference, we are now able to perform them in more acceptable times, even with an important number of parents.

Table 2: Decomposition and inference time to compute city's water level and size of the generated file, depending on $|\,\text{Pa}\,|$

$\lvert\,\text{Pa}\,\rvert$	decomposition	inference	file size
5	3e-4s	6e-3s	103Ko
25	2e-4s	6e-1s	11Mo
50	1e-3s	5s	85Mo
100	1e-2s	71s	673Mo

5 Conclusion

In this paper, we have proposed a new model of probabilistic production rules that responds to an industrial need for a simplified syntax, at the cost of adding a layer of complexity in the compilation phase of a rule engine, partially due to the requirements of high interoperability with the probabilistic engine. The technical contribution is implemented as a plugin used in a modern BRMS and don't have, to the extent of our knowledge, any element of comparison.

Subsequently, we defined a way to make self-decomposable probabilistic aggregator usable in a context where inferences are expected to be fast and reliable. We will try, as a future work, to find ways to embed large factors into compact representations in order to reduce inference time in incrementally changing large-scale worlds.

Acknowledgments. This work was supported by IBM France Lab/ANRT CIFRE grant #2018/0251

References

Agli, H. 2017. *Uncertain reasoning for business rules*. Ph.D. Dissertation, Université Pierre et Marie Curie - Paris VI.

Buchanan, B., and Shortliffe, E. 1984. *Rule-based Expert System – The MYCIN Experiments of the Stanford Heuristic Programming Project*.

Cooper, G. F. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence* 42(2):393 – 405.

Forgy, C. L. 1982. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence* 19(1):17 – 37.

Hart, P. E.; Duda, R. O.; and Einaudi, M. T. 1978. Prospector-a computer-based consultation system for mineral exploration. *Journal of the International Association for Mathematical Geology*.

Heckerman, D., and Shortliffe, E. 1992. From certainty factors to belief networks. *Artificial Intelligence In Medicine*.

Jesus, P.; Baquero, C.; and Almeida, P. 2015. A survey of distributed data aggregation algorithms. *IEEE Communications Surveys and Tutorials* 17(1):381–404.

Koller, D., and Pfeffer, A. 1997. Object-oriented bayesian networks. *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)* 302–313.

Madsen, A. L., and Jensen, F. V. 1999. Lazy propagation: a junction tree inference algorithm based on lazy evaluation. *Artificial Intelligence*.

Medina Oliva, G.; Weber, P.; Levrat, E.; and Iung, B. 2010. Use of probabilistic relational model (prm) for dependability analysis of complex systems. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*.

Ng, K. C., and Abramson, B. 1990. Uncertainty management in expert systems. *IEEE Expert-Intelligent Systems and their Applications*.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems: networks of plausible inference (Morgan kaufmann series in representation and reasoning)*. Morgan Kaufmann Publishers, San Mateo, Calif.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1-2):107–136.

Robertson, N., and Seymour, P. 1986. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms* 7(3):309 – 322.

Torti, L.; Wuillemin, P.-H.; and Gonzales, C. 2010. Reinforcing the object-oriented aspect of probabilistic relational models. In *Proceedings of the 5th European Workshop on Probabilistic Graphical Models, PGM 2010*.

Weber, P.; Medina-Oliva, G.; Simon, C.; and Iung, B. 2012. Overview on bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence*.

Zadeh, L. 1965. Fuzzy sets. *Information and Control* 8(3):338–353.