Advanced Syntax and Compilation for Probabilistic Production Rules with PRM

Gaspard Ducamp, Philippe Bonnard, Christian de Sainte Marie, Pierre-Henri Wuillemin

Advanced Syntax and Compilation for Probabilistic Production Rules with PRM

Gaspard Ducamp[1,2], Philippe Bonnard[2], Christian de Sainte Marie[2], and Pierre-Henri Wuillemin[1]

[1] LIP6 (UMR 7606), Sorbonne Université, 4 place Jussieu, 75005 Paris, France
prenom.nom@lip6.fr
[2] IBM France Lab, 9 rue de Verdun, 94250 Gentilly, France
philippe.bonnard@fr.ibm.com, csma@fr.ibm.com, gaspard.ducamp@ibm.com

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 syntax for PPR making their use easier for business users and showcase how we managed to adapt to the compilation toolchain of an industrial rule engine accordingly.

Keywords: Uncertain reasoning · Business Rules · Probabilistic Relational Models · Bayesian Networks

1 Context

Business Rules Management Systems (BRMS), such as *IBM Operational Decision Manager* (ODM), are developed since the 90's to facilitate authoring, testing, 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. When developing intelligent systems, it may be inevitable to deal with uncertainty. This issue can have multiple origins such as measurement errors, noisy automatic process or even the modeling process itself. Handling such uncertainty in a BRMS could allow business user to represent and reason with complex and real-world data.

Numerous methods have been used in the rule-based system community to deal with uncertainty, using *certainty factors* [3], *likelihood ratio* (Hart et al. [8]) or even *fuzzy logic* [20]. However, there was some limitations using such approaches, mainly due to interpretation being incoherent with probability theory [9] or inconsistency in the conclusions when performing chains of inference [13].

Bayesian techniques, mostly based of Bayesian networks [14, 19], have been used to model domains with uncertainty but are not suited for complex systems involving high design and maintenance costs [11]. Another solution could be to use models that combine first-order logic and probabilistic reasoning, such as Markov logic networks [15], but their abstract structure is incompatible with business rules' principles. The following results are part of a PhD thesis whose primary ideas where discussed in [5].

This paper will start with a brief 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 increase their expressivity using a tight coupling of a BRMS with an object-oriented PGM. 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 Uncertainty with 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 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 Figure 1.a.

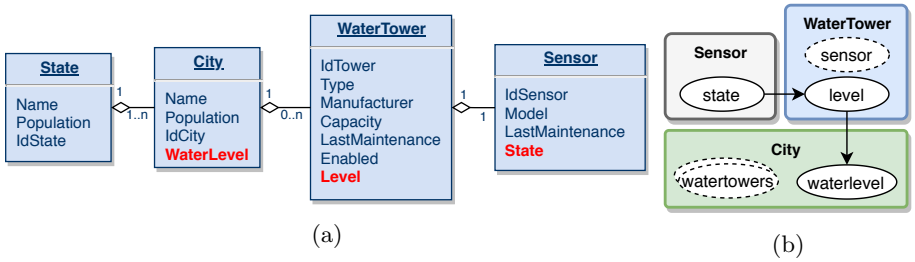


Fig. 1: (a) Class diagram for the water shortage example (the hollow diamond shape indicates that a state 'has' at least one city and a city is only in one state). Red variables are probabilistic as opposed to blue ones whose values are always deterministic (certain). (b) PRM class dependencies schema

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 [7, 17]. The rule presented in Table 1, for example, is used to identify broken sensors inside water towers, allowing technicians to be alerted when maintenance is required. However one would like to add uncertainty in rules, for instance in order to perform predictive maintenances rather than corrective ones, reducing the risk of unexpected breakdowns. Previous work [2, 1] showed

that a loose coupling between a rule engine and probabilistic graphical models (Bayesian networks initially, then probabilistic relational models) allowed reasoning and making decision with uncertain data. However, a number of problems have been raised with those approaches [5], mainly related to accessibility issues for a business user, the proposed syntaxes requiring a deep understanding of the probabilistic model used. In particular, they require business user to define separate decision thresholds on each separate uncertain variable. For instance, to define a rule that represents the policy of water usage being restricted under certain conditions of temperature and water stock level, the user has to explicit the minimal probability of the temperature reaching the specified policy threshold and the minimal probability of the stock being below the specified policy threshold; and so on for each probabilistic variable in each condition of each rule.

Table 1: Example of rule used in a BRMS

```
rule SensorMaintenance {
  when {
        c: City() ;
        wt: WaterTower() in c.watertowers;
        s: Sensor(s.state==broken) from wt.sensor;
  } then {change the sensor}}
```

A Bayesian network (BN) is a compact representation of a joint probability distribution over a set of random variables. These appear in the form of nodes in a direct acyclic graph (DAG) where the absence of arcs represent conditional independences. Each node is associated with a conditional probability table (CPT) that contains the conditional probabilities of the random variable with respect to its parents. As said before, 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 [18], where the focus is set on classes of objects and by defining relations among them. The expressiveness gained when adding notions of random variables and conditional probabilities to classes, attributes, relations, interface, inheritance and instantiations makes graphical models reusable and scalable [12]. 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 1.b shows an example of relation schema for PRM classes generated from the model described in Figure 1.a. 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 but depending on the model, the date of its last maintenance and its state, the sensor will work within a certain level of confidence, hence the uncertainty.

3 A new definition of PPR

To address the business user friendliness issue raised above, we have redefined the treatment of uncertainty in the expression of rules by replacing the probability thresholds attached to single variables by an aggregated 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 allows our probabilistic rules to be more accessible but it required a redefinition of the rules compilation phase to redistribute the overall risk to each individual condition. In the example in Table 2, a city will restrict its water usages if there is a high probability that either its temperature is high or its water resources are lower than 10^4 megalitres.

Table 2: A new syntax for probabilistic rule

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

To achieve this, it is necessary to generate the PRM not only from the defined object model but also from the set of rules. Since the rule engine does not know how to 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.

3.1 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 2), then (i) the predicates present in the conditions are added to the class, as well as the operators that connect them (in our case an *or*); (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.

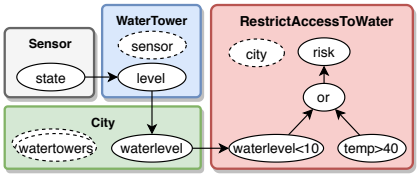


Fig. 2: Class dependencies schema of the enhanced PRM

Because conditions are henceforth encoded in our PRM, the rule evaluation will be based on the use of the probabilistic engine, reinforcing the coupling between engines. Once the PRM enhanced with the new classes, 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. A rewritten rule will have the following form:

Table 3: Rewritten rule

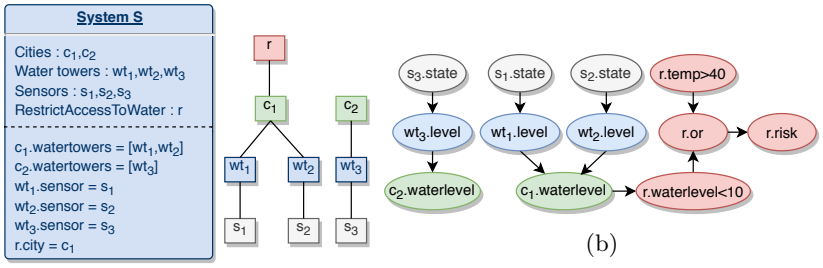
```
rule RestrictAccessToWater {
  when {
       c: City ();
       evaluate (PRMengine.getRisk("WaterShortageMode", [c],
                                        [c.temperature>40]  ) > 0.9) ;
  } then {restrict water usage to key functions}}
```

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, as shown in Table 3: (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 (in our case a city); (iii) The value of the deterministic elements encoded in the class. Since the city c is known by the rule engine, we can certainly verify the truth value of the predicate ($c.temperature > 40$). These values will be used as evidences in the system.

3.2 Runtime, PRM system and working memory

In order to be able to work with a PRM, we need to define its components (with the PRM model) but also to instantiate them. To do so, we are 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 3 shows such a system in a case where the working memory contains only 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 $c_1.temperature > 40$.

From a relation skeleton we can generate a BN called *grounded BN*, as shown on Figure 3.b; we create a node for each attributes of the objects in the relation skeleton and linked them according to the dependencies in the PRM model. Once the grounded BN generated, we can compute the posterior of the $r.risk$ node given all the evidences.

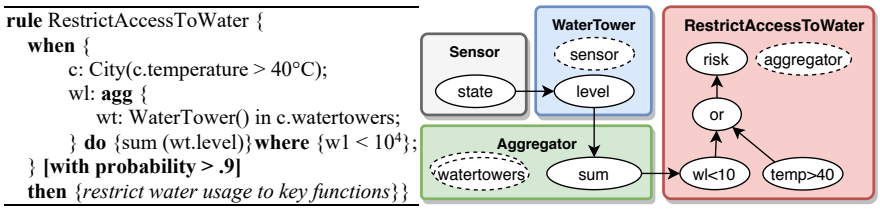


(a) (b)

Fig. 3: (a) Relational skeleton based on Fig. 2 (b) Grounded BN from the skeleton

3.3 Probabilistic aggregation function

If the aggregation is no longer defined in the model but as a condition in the rule, like in the example in Figure 4.a, the compiler detects its structure and translate it into a PRM aggregation, the rule is rewritten accordingly. The PRM model is enhanced with the addition of a class representing the rule (*RestrictAccessToWater*) as well as with a class containing the aggregator (*Aggregator.sum*). The *evaluate* instruction of the rewritten rule will be called with a list of water towers instead of a city in order to instantiate references in consequence.



(a) (b)

Fig. 4: (a) Example of probabilistic rule using an aggregation function. (b) Class dependencies schema of the enhanced PRM

3.4 Implementation

This development work was based on what was done in the Bayesian Insight Service (BIS) plugin developed in [1]. Figure 5 illustrates how our new module, PRIME (Probabilistic Reasoning Insight ModulE), fits into the ODM's toolchain. It intervenes directly during the process of rewriting the semantic tree describing the rules (*SemRuleset*) but, unlike BIS, extends the definition and optimization of the graphical model from the rules before rewriting them

(*PRM enhancement* process). Once the rules are rewritten, the graphical model is serialized in order to be usable by a probabilistic engine in parallel with ODM, in our case aGrUM (https://agrum.gitlab.io). At runtime, an API is used for PRIME to interoperate with aGrUM, in order to request probabilistic values keep the PRM system up-to-date (if a sensor is replaced, for example).

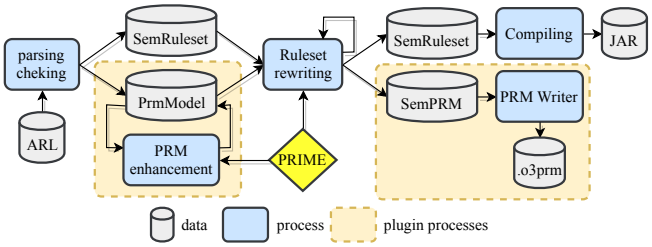


Fig. 5: ODM's toolchain using the PRIME plugin

4 Current works

Since this work, emphasis has been placed on the performance of inferences in PGM. When performing an inference based on message-passing algorithm [10], we use a secondary structure called a Junction Tree where variables are grouped into cliques according to their parents. The complexity of inference in a BN is NP-Hard [4], growing exponentially in the tree-width of the network, the tree-width being related to the size of its largest clique (determined by the products of the domains of its variables) [16]. This is one of the major issues when dealing with probabilistic aggregators, especially when they have a high number of parents since there will be at least a clique of the size of this family.

In order to make the computation of posteriors possible we have proposed a first approach based on aggregator decomposition [6]. Computing the distribution of $c1.waterlevel$ in Figure 3.b but for a city with 7 water towers (each taking up to 10 values) would have needed at least $70 \cdot 10^7$ values to be stored and multiple hours of calculation. By a simple manipulation of the structure of the BN before the inference, we managed to reduce the number of parameters to store to $\approx 40 \cdot 10^3$ and computation time to less than a second. This transformation makes inferences scalable and usable in an industrial context, but only works with a certain type of aggregation functions. For this reason, we are currently working on a new approximate inference capable of working with complex networks, regardless of the nature of the nodes which compose them.

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

References

1. Agli, H.: Uncertain reasoning for business rules. Ph.D. thesis, Université Pierre et Marie Curie - Paris VI (2017)
2. Ait-Kaci, H., Bonnard, P.: Probabilistic production rules. Tech. rep., IBM (2011)
3. Buchanan, B., Shortliffe, E.: Rule-based Expert System – The MYCIN Experiments of the Stanford Heuristic Programming Project (01 1984)
4. Cooper, G.F.: The computational complexity of probabilistic inference using bayesian belief networks. Artificial Intelligence (1990)
5. Ducamp, G., Bonnard, P., de Sainte Marie, C., Gonzales, C., Wuillemin, P.H.: Improving probabilistic rules compilation using prm. In: RuleML+RR Doctoral Consortium 2018 (2nd International Joint Conference on Rules and Reasoning). Esch-sur-Alzette, Luxembourg (2018)
6. Ducamp, G., Bonnard, P., Wuillemin, P.H.: Uncertain reasoning in rule-based systems using prm. In: Florida Artificial Intelligence Research Society Conference (2020)
7. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence (1982)
8. Hart, P.E., Duda, R.O., Einaudi, M.T.: Prospector-a computer-based consultation system for mineral exploration. Journal of the International Association for Mathematical Geology (1978)
9. Heckerman, D., Shortliffe, E.: From certainty factors to belief networks. Artificial Intelligence In Medicine (1992)
10. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)
11. Koller, D., Pfeffer, A.: Object-oriented bayesian networks. Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97) (1997)
12. Medina Oliva, G., Weber, P., Levrat, E., Iung, B.: Use of probabilistic relational model (prm) for dependability analysis of complex systems. In: IFAC Proceedings Volumes (IFAC-PapersOnline) (2010)
13. Ng, K.C., Abramson, B.: Uncertainty management in expert systems. IEEE Expert-Intelligent Systems and their Applications (1990)
14. Pearl, J.: Probabilistic reasoning in intelligent systems: : networks of plausible inference (Morgan kaufmann series in representation and reasoning). Morgan Kaufmann Publishers, San Mateo, Calif. (1988)
15. Richardson, M., Domingos, P.: Markov logic networks. Machine learning **62**(1-2), 107–136 (2006)
16. Robertson, N., Seymour, P.: Graph minors. ii. algorithmic aspects of tree-width. Journal of Algorithms **7**(3), 309 – 322 (1986)
17. Silva, B.: Verification of Business Rules Programs. Ph.D. thesis, Albert-Ludwigs-Universität Freiburg (2012)
18. Torti, L., Wuillemin, P.H., Gonzales, C.: Reinforcing the object-oriented aspect of probabilistic relational models. In: Proceedings of the 5th European Workshop on Probabilistic Graphical Models, PGM 2010 (2010)
19. Weber, P., Medina-Oliva, G., Simon, C., Iung, B.: Overview on bayesian networks applications for dependability, risk analysis and maintenance areas. Engineering Applications of Artificial Intelligence (2012)
20. Zadeh, L.: Fuzzy sets. Information and Control **8**(3), 338–353 (1965)