# Probabilistic Perception Revision in AGENTSPEAK(L)

Francisco Coelho and Vitor Nogueira

Departamento de Informática, Escola de Ciências e Tecnologia, Universidade de Évora
{fc,vbn}@di.uevora.pt

**Abstract.** Agent programming is mostly a symbolic discipline and, as such, draws little benefits from probabilistic areas as machine learning and graphical models. However, the greatest objective of agent research is the achievement of autonomy in dynamical and complex environments — a goal that implies embracing uncertainty and therefore the entailed representations, algorithms and techniques. This paper proposes an innovative and conflict free two layer approach to agent programming that uses already established methods and tools from both symbolic and probabilistic artificial intelligence. Moreover, this method is illustrated by means of a widely used agent programming example, GoldMiners.

## 1 Introduction and Motivation

Agent autonomy is a key objective in Artificial Intelligence (AI). Complex environments, like the physical world where robots must delve, impose a degree of uncertainty that challenges symbolic processing. But while a probabilistic approach, currently expressed in Machine Learning (ML) and Probabilistic Graphical Models (PGMs) [14], is required for certain aspects of autonomy, a great deal of agent programming is better handled by declarative programming (*e.g.* PROLOG) and more specifically, Beliefs, Desires and Intentions (BDI) architectures for autonomous agents, part of symbolic AI.

Symbolic and probabilistic areas of AI are not necessarily incompatible. Consider for example distribution semantics [18] and markov logic [8]. From there two paths exist towards the interplay of symbolic and probabilistic AI: extending PGMs with logical representations, in Statistical Relational Learning (SRL) [18], and extending logic programming languages with probability, in Probabilistic Logic Programming (PLP) [11,12]. For autonomous agents the symbolic *vs.* probabilistic division persists. Symbolic architectures, such as BDI, describe agent behavior on the basis of metaphors (*e.g.* goals, beliefs, plans) drawn from human behavior while the principle of Maximum Expected Utility (MEU) guides probabilistic AI but there is only seminal work blurring that division.

Concerning agents programming JASON [6] is a popular AGENTSPEAK(L) (ASL) [17] interpreter and framework, triggering a considerable amount of research (*e.g.* [5]). The BDI architecture in general, including ASL and JASON in particular, outline a set of symbolic data structures and processes with more or less detailed semantics. However we can see these agents in trouble when their environment becomes stochastic. This assertion is supported by the experiment plotted in Figure 3: the GoldMiners (GM) is a virtual scenario used in the 2006 Multi-Agent Programming Contest [3] edition, now part of JASON's examples. The two playing teams reach scores that are clearly

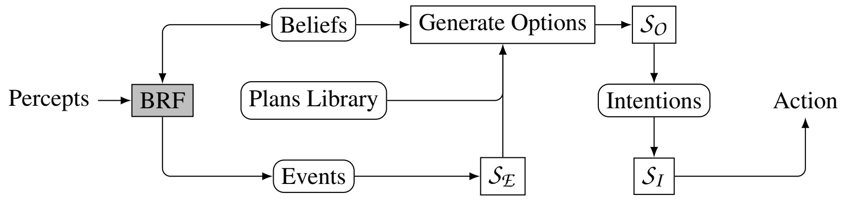

**Fig. 1.** The JASON deliberation process outlined, with the BRF highlighted. Percepts are processed to generate events and update the beliefs base. Available options are instantiated plans triggered by one event (selected by $\mathcal{S}_\mathcal{E}$) and compatible with the current beliefs. One option (defined by $\mathcal{S}_O$) is then appended to the intentions, where $\mathcal{S}_I$ chooses an action.

reduced even with a small amount of sensor misreadings. It turns out that Bayesian Networks (BNs) are representations of dependency of random variables and, thus, natural candidates to represent probabilistic beliefs. But replacing symbolic beliefs by BNs is not trivial in part because changing the symbolic nature of beliefs entails reconsiderations about the BDI architecture (e.g. the beliefs base must unify with plans contexts; changing these to a distribution will break the semantic of such unifications).

Our proposal is, at large, to wrap a layer of probabilistic techniques around certain symbolic processes without altering those processes or associated semantics. Here we illustrate this approach by focusing on the perception. In a stochastic environment, with a certain probability, values reported by sensors differ from the actual value. If sensor reported values are directly used by the deliberation process then performance suffers a penalty that results from the illusions about the truth of the environment. But sensor misreadings can be partially corrected under certain conditions using probabilistic methods. Our task is to find out if with such corrections performance degradation is attenuated and the added complexity has little impact in the deliberation cycle.

The remainder of this paper is organized as follows: next are provided the main concepts of these areas, followed, in section 2, by a general description of the percept-correction function (PCF) in the BDI architecture and a specific instantiation for the GM scenario extended with sensor misreadings. Section 3 presents a particular experiment on that scenario and respective results. In the last section the authors draw some conclusions on that experiment and outline future research.

## 1.1 State of the Art

Here we outline the groundwork of our proposal: the ASL language, its interpreter JASON, together with PGMs, Dynamic Bayesian Networks (DBNs), and Hidden Markov Models (HMMs).

**AGENTSPEAK(L) and JASON.** BDI is the predominant architecture used for defining intelligent agents. ASL [17] can be described as a logic programming based language geared towards the BDI architecture. JASON [6,4] implements the operational semantics of an extension of ASL and its deliberation cycle is depicted in Figure 1. In this cycle, the environment generates percepts that are processed by a belief-revision function (BRF). Each change in the beliefs base generates an event. Goals in the set of events represent different desires that the agent selects by the function $\mathcal{S}_{\mathcal{E}}$. The selected event entails applicable plans (options) instantiated from the plans library. Selection of a plan among applicable ones is performed by the function $\mathcal{S}_O$ and included in the set of (current) intentions. Finally function $\mathcal{S}_I$ selects one action from the set of intentions the one (action). Although the BRF evaluation is not part of the ASL specification it is a necessary component of the architecture. The default one that comes with JASON "simply updates the belief base and generates the external events in accordance with current percepts. In particular, it does not guarantee belief consistency." [4] (nevertheless, in [1], the authors present a polynomial-time belief revision algorithm that restores belief base consistency when there are derived inconsistencies). JASON is used as the ASL framework and scenario simulator in this work.

**Hidden Markov Models and Dynamic Bayesian Networks.** HMM is a well-known framework to deal with latent variables in stochastic processes [2,16,14]. A (discrete) system state at time step $t$ is described by a random variable $X^{(t)}$ that verifies the markov condition $\Pr\left(X^{(t+1)} \mid X^{(0:t)}\right) = \Pr\left(X^{(t+1)} \mid X^{(t)}\right)$. This distribution is the *transition model* of the system. If $X^{(t)}$ is hidden but a *sensor model* $\Pr\left(Y^{(t)} \mid X^{(t)}\right)$ is known then the *filter problem* is to compute the distribution of $X^{(t)}$ given an *initial state*, $x^{(0)}$, and a *sequence of observations*, $y^{(1:t)}$. The *forward* algorithm is a common procedure to compute $\Pr\left(X^{(t)}\right)$ that requires a belief about the previous environment state, $\hat{x}^{(t-1)}$, updates it with the transition model and corrects that update with the sensor model and current sensor reading, $y^{(t)}$. The major problem with a naïve approach of HMMs is that the size of the transition model is quadratic in the number of system states. DBNs tries to minimize this problem by exploiting independences in the *structure* of the system to, hopefully, produce smaller representations of the transition and sensor models. The general HMM and DBN frameworks can be used directly to describe agent related problems. Perceptions are represented by the observation model and actions by an observed variable, say $a^{(t)}$. The transition model becomes $\Pr\left(X^{(t)} \mid X^{(t-1)}, A^{(t)}\right)$ and the sensor model $\Pr\left(Y^{(t)} \mid X^{(t)}, A^{(t)}\right)$. DBN are used in this work to correct agent perceptions. In this specific case the transition and sensor models have many independence relations that are exploited to obtain "small" matrices. The next section describes the construction of such models.

## 2 Probabilistic Perception Correction

Here we illustrate how probabilistic methods can be used to improve performance in the GM noisy scenario without changing original the symbolic processing.

Currently the problem of extending ASL with probabilistic features is mostly directed to belief representation and addressed by many authors [15,10,9,13,19] but isn't yet fully solved. An alternative and less intrusive application of probabilistic AI to ASL, proposed here, targets the processes instead of the data structures. Bounding probabilistic techniques to the computation of certain ASL functions (e.g. the BRF, $\mathcal{S}_{\mathcal{E}}, \mathcal{S}_O, \mathcal{S}_I$ functions, as in figure 1) promises a number of advantages. Since the computations of those functions are unspecified in ASL (and overwritable in JASON), probabilistic techniques can be used there without invalidating previous work. So symbolic and prob-

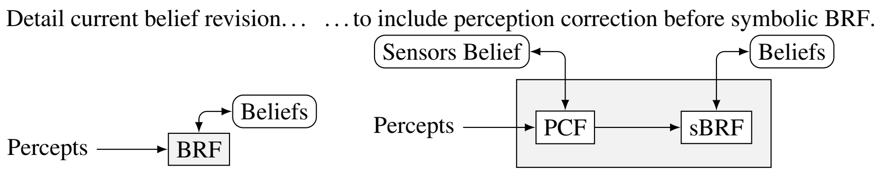
Detail current belief revision. . .
. . . to include perception correction before symbolic BRF.
Percepts
BRF
Beliefs
Sensors Belief
Percepts
PCF
sBRF
Beliefs

**Fig. 2.** Inclusion of percept-correction function (PCF) before the original, symbolic, BRF (denoted by sBRF) to correct noisy perceptions. The Sensors Belief is a distribution of sensor values and independent of the (symbolic) beliefs used in the BDI deliberation.

abilistic AI have clearly separated roles and each is used to solve "familiar" problems in the respective domain while both simultaneously contribute to the agent behavior. Symbolic programming uses unchanged ASL programs to define high level agent behavior, with plans, beliefs, *etc.* while probabilistic algorithms process low level noisy signals — with BNs, influence diagrams, *etc.*

Defining certain functions of ASL as tasks to be solved by probabilistic techniques seems a promising technique to address open problems in agent autonomy, using already known theory and tools. Next we describe the setup of an example of this interplay of symbolic and probabilistic techniques. This experiment uses original ASL programs designed for a (mostly) deterministic scenario; adds sensor misreadings, the respective rate being defined by a parameter; re-defines the computation of the BRF with the help of a probabilistic process and records the performance of agents. The outcome of the first two steps is depicted in Figure 3 where performance degradation is associated with increased sensor misreadings.

**Problem Statement: (Noisy) GoldMiners.** The GM competition is described in [7]. A miner is equipped with a $3 \times 3$ grid of sensors, $Y_{0:8}$, that scans its neighborhood. Each sensor reports the content of a cell, that can be one of *empty*, *obstacle*, *gold* or *miner*. The miner can also select one action of *up*, *down*, *left*, *right*, *pick*, *drop* and *skip*. Non-determinism is present as incomplete perception and action failure and in the examples in JASON noise increases the probability of action failure in proportion to current cargo. Values depicted in Figure 3 result from the JASON simulator with noisy sensors. The "noise" parameter is the rate of cell misreadings. Sensors are independent but equally parameterized: the value reported by each sensor depends only on the noise parameter and cell content.

**Proposal: Percept-Correction Function.** Our proposal to recover agent performance is to prepend a percept-correction function (PCF) to the BRF applying probabilistic knowledge of the environment (see Figure 2). Correction of perceptions is an inference problem in the framework of HMMs and DBNs .

The formal problem statement is: *In the GM simulation extended with sensor noise parameter* $\theta$*, update the estimate of cells content* $\hat{x}'_{0:8}$ *given the previous estimate* $\hat{x}_{0:8}$*, current action* $A' = a'$ *and sensor readings* $Y'_{0:8} = y'_{0:8}$. For notation simplicity we write $X = X^{(t-1)}_{0:8}, X' = X^{(t)}$, *etc.* A resolution is as follows. Given the previous estimate $\hat{x}$,

| | | |
|---|---|---|
| $N(0)=\{0,1,3\}$ | $N(1)=\{0,1,2,4\}$ | $N(2)=\{1,2,5\}$ |
| $N(3)=\{0,3,4,6\}$ | $N(4)=\{1,3,4,5,7\}$ | $N(5)=\{2,4,5,8\}$ |
| $N(6)=\{3,6,7\}$ | $N(7)=\{4,6,7,8\}$ | $N(8)=\{5,7,8\}$ |

**Table 1.** Neighbors in the grid sensor. The function $N(i)$ defines the set of neighbors of sensor $i$. From this topology follow independence statements of the form $X'_i \perp\!\!\!\perp X_{\setminus N(i)} \mid X_{N(i)}$ where $\setminus N(i)$ is a short-hand to $1:8 \setminus N(i)$.

current action $a'$ and sensor reading $y'$, the update is

$$\Pr(X' \mid \hat{x}, a', y') \propto \Pr(y' \mid X') \Pr(X' \mid \hat{x}, a') \tag{1}$$

and perception correction is the *maximum a posteriori* (MAP) of each sensor,

$$\hat{x}'_i = \arg_s \max \Pr(X'_i = s \mid \hat{x}, a', y') \tag{2}$$

where $s$ ranges over all (four) sensor values. The $\arg_s \max$ computation doesn't require normalizing the right side of Equation 1, a welcome simplification but direct calculation doesn't scale to the grid sensor. Each location has four different values so the state space has $4^9$ and for each action the corresponding transition has $4^{9\times 2}$ parameters. Numbers of this magnitude render intractable a direct HMM approach. Fortunately, the grid sensor entails many independence statements that reduce the number of transition and observation parameters to a convenient size:

1. Sensor values are independent between them so instead of a “big” transition we only need to deal with nine “small” transitions, one for each sensor;
2. Updated values of each sensor depend only on the action and the previous values of neighbor sensors, defined in Table 1. This can be further refined when the action is considered (e.g. for the “up” action the “bottom” neighbors are irrelevant);
3. The observed value of a sensor depends only on the corresponding cell content;

Using these independence statements the transition and observation models can be described by relatively small matrices and the computations of Equations 1 and 2 become acceptable for the inclusion of the PCF in the BDI deliberation process. The sensor model is very simple:

$$\Pr(Y'_i = x \mid X'_i = x) = 1 - \theta, \forall x \tag{3}$$

for noise parameter $\theta$. The transition model, more complex than the sensor model, is explained in the next sub-section.

**Resolution: Transition Model.** The transition model is based on a few simplifying assumptions about the environment:

1. The state of the environment only changes by effect of the miner's actions. In particular detected miners do not move and gold doesn't “appear” in empty cells;
2. The miner never moves to “obstacle” or “miner” cells;
3. The content of unscanned cells is uniformly distributed over all possible values;

The state of a sensor depends only on the action and previous values of the neighbors. Different actions entail different schemes for the transition parameters, easier to describe one action at a time.

**Action "skip".** In this case the miner doesn't change the environment state;

**Action "pick".** The miner removes a gold from its location, if one exists;

**Action "drop".** The miner adds a gold in its location, if that cell is empty;

**Action "up".** The miner moves up and the sensors that enter unscanned cells are $\{0,1,2\}$. For these cells the scanned value is uniformly distributed. Each one of the other sensors scans the cell previously above it;

**Remaining actions ("right", "down" and "left".)** These are similar to "up";

These descriptions can be easily translated into conditional probabilities that completely define the transition model for the sensor grid. That model can be used by the forward algorithm outlined before and Equations 1 – 2 to correct perceptions.

## 3 Results

The proposed approach requires support on the effect it may have on agents performance. Our option was to gather empirical evidence to guide further research.

**Implementation Notes.** JASON already provides a simulator for the original GM. Its object-oriented nature helps the construction of a noisy variant by overriding some classes and methods. Computation of the PCF uses two libraries developed by the authors, one (at https://github.com/fmgc/jpgm) targeted to the sparse representation of matrices and related operations and a second one (at https://github.com/fmgc/ngm) bridging the operations of the first library to the noisy GM scenario. These are minimal libraries providing only the necessary support here.

**Empirical Results.** Evaluation of the effect of the PCF uses three teams: a "dummy" team that barely uses any ASL features; a "smart" team that makes heavy use of ASL but not the PCF and a "corrected" team that has the same ASL programs of the "smart" team and uses the PCF. Performance of each team is the number of gathered golds after a given number of time-steps and the sensor noise rate is set at five values: $0.000, 0.025, 0.050, 0.075$ and $0.100$. A run is defined by a team and a noise value and simulates the GM scenario for 700 time-steps. At the end the number of gathered golds is recorded. Each run (team, noise) is launched ten times and those runs are summarized by the mean and standard variation of the number of gathered golds. The final result are fifteen pairs of (mean, standard variation), plotted in Figure 3.

## 4 Conclusion

These results provide empirical support to further research the interplay of symbolic and probabilistic AI specifically concerning BDI agent architectures.

There seems to be no major *theoretical* obstacles to generalize this method to other domains and applications. However probabilistic inference in general is intractable [2] despite factorization methods and efficient algorithms for specific graph structures (e.g. DBN, Junction Tree). Probabilistic *learning* was not considered here, although *unsupervised bayesian learning* seems tailored to suit autonomous agents since the benefits of self-reconfiguration might prove critical. Again, adoption of such features stumbles into the computational complexity of the problem.

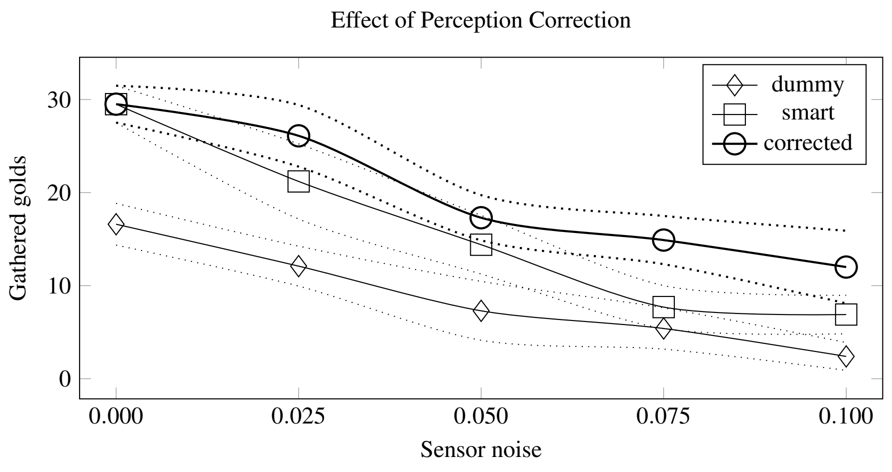

**Fig. 3.** Sensor noise (horizontal axis) *vs.* agent performance measured by gathered golds (vertical axis). Performance of the "dummy", "smart" and "corrected" teams under various levels of sensor noise are plotted. Each data point summarizes the number of gathered golds by team in a given noise value and consists of the mean (black line) and standard variation (band of dotted lines) of ten samples. The results of the "corrected" team are clearly above the others.

For relatively simple scenarios the definition of the transitions of the probabilistic model of the environment used by the PCF can be done "by hand" but for problems with large number of variables this raises an usability problem with no easy resolution. This is also an issue with probabilistic methods, not specific to this work.

One particular problem with this symbolic/probabilistic separation is that there can be inconsistencies between the probabilistic model and the symbolic beliefs. Also the advantages of a single, coherent and theoretically sound language cannot be easily discarded. Seemingly in opposition to this line of research, an unified symbolic and probabilistic framework using Markov Logic (MkL), Statistical Relational Learning (SRL) or Probabilistic Logic Programming (PLP) for example, could, in principle, simplify the semantic study of agent behavior and formal verification of agent programs.

**Future Work.** The major application area seems to include robotics where intrinsically noisy perception is one of the major obstacles to symbolic controls. Hopefully this line of research might facilitate such integration. Further development of this work folds into four major lines: *formal specification and semantics* to support verification of agent programs, guaranteed behavior, *etc*; *further applications of probabilistic methods* to symbolic agent programming (*e.g.* use of influence diagrams to sort actions in the intention selection); *simulations* in virtual environments is a key step in robotics. JASON already provides a large set of scenarios ready to explore in the lines of the GM example presented here; *deployment in physical robots* like the ardrone or twopi is a major challenge given the (usual) computation and real-time constraints of such platforms.

## Acknowledgements

The people around us, the flow of life, the internet. And, of course, the gracious money provider, Fundação para a Ciência e Tecnologia.

## References

1. Alechina, N., Bordini, R.H., Hübner, J.F., Jago, M., Logan, B.: Belief Revision for AgentSpeak Agents. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems. AAMAS '06 (2006)
2. Barber, D.: Bayesian reasoning and machine learning. Cambridge University Press (2012)
3. Behrens, T., Dix, J., Köster, M., Hübner, J.: Special issue about multi-agent-contest II. Annals of Mathematics and Artificial Intelligence 61 (2011)
4. Bordini, R.H., Hübner, J.F.: BDI agent programming in AgentSpeak using Jason. In: Computational logic in multi-agent systems, pp. 143–164. Springer (2006)
5. Bordini, R.H., Hübner, J.F.: Semantics for the Jason variant of AgentSpeak (plan failure and some internal actions). In: ECAI. pp. 635–640 (2010)
6. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. Wiley (2007)
7. Dastani, M., Dix, J., Novák, P.: The second contest on multi-agent systems based on computational logic. In: Computational Logic in Multi-Agent Systems. Springer (2007)
8. Domingos, P., Kok, S., Poon, H., Richardson, M., Singla, P.: Unifying logical and statistical AI. In: AAAI. vol. 6, pp. 2–7 (2006)
9. Fagundes, M.S.: Integrating BDI model and Bayesian Networks. Master's thesis, Universidade Federal do Rio Grande do Sul (2007)
10. Fagundes, M.S., Vicari, R.M., Coelho, H.: Deliberation process in a BDI model with bayesian networks. In: Agent Computing and Multi-Agent Systems. pp. 207–218. Springer (2009)
11. Fierens, D., den Broeck, G.V., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., Raedt, L.D.: Inference and learning in probabilistic logic programs using weighted boolean formulas. arXiv p. 1304.6810 (04 2013), `http://arxiv.org/abs/1304.6810`
12. Gutmann, B., Thon, I., De Raedt, L.: Learning the parameters of probabilistic logic programs from interpretations. In: Machine Learning and Knowledge Discovery in Databases, pp. 581–596. Springer (2011)
13. Kieling, G.L., Vicari, R.M.: Insertion of probabilistic knowledge into BDI agents construction modeled in bayesian networks. In: Complex, Intelligent and Software Intensive Systems (CISIS), 2011 International Conference on. pp. 115–122. IEEE (2011)
14. Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. The MIT Press (2009)
15. Luz, B., Meneguzzi, F., Vicari, R.: Alternatives to threshold-based desire selection in bayesian BDI agents. In: 1st International Workshop on Engineering Multi-Agent Systems (2013)
16. Murphy, K.P.: Machine learning: a probabilistic perspective. MIT press (2012)
17. Rao, A.S.: AgentSpeak (L): BDI agents speak out in a logical computable language. In: Agents Breaking Away, pp. 42–55. Springer (1996)
18. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: Proceedings of the 12th International Conference on Logic Programming (ICLP'95) (1995)
19. Silva, D.G., Gluz, J.C.: AgentSpeak(PL): A new programming language for BDI agents with integrated bayesian network model. In: International Conference on Information Science and Applications (2011)