Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities

Dominik Jain, Lorenz Mösenlechner and Michael Beetz
Intelligent Autonomous Systems, Technische Universität München
{jain, moesenle, beetz}@cs.tum.edu

Abstract— **An autonomous robot system that is to act in a real-world environment is faced with the problem of having to deal with a high degree of both complexity as well as uncertainty. Therefore, robots should be equipped with a knowledge representation system that is able to soundly handle both aspects. In this paper, we thus introduce an architecture that provides a coupling between plan-based robot controllers and a probabilistic knowledge representation system based on recent developments in statistical relational learning, which possesses the required level of expressiveness and generality. We outline possible applications of the corresponding models in the context of robot control, discussing suitable representation formalisms, inference and learning methods as well as transparent extensions of a robot planning language that allow robot control programs to soundly integrate the results of probabilistic inference into their plan generation process.**

I. Introduction

Autonomous robots that are to act in human environments, which lack determinism by definition, must be equipped with powerful probabilistic reasoning mechanisms in order to make reasonable assumptions about the (future) state of their inherently dynamic and stochastic environment. Not only is probabilistic reasoning essential in solving high-dimensional state estimation problems as well as in achieving robust control programs by allowing the anticipation of likely failures and enabling a choice of actions that will most probably succeed; it is also the only way to enable an autonomous robot to draw meaningful conclusions about the complex interactions between entities (in particular, humans) in its environment, which are indispensable in human-robot interaction scenarios. Therefore, probabilistic reasoning is required not only at a subsymbolic level, but also at the symbolic level, which we advocate in this paper.

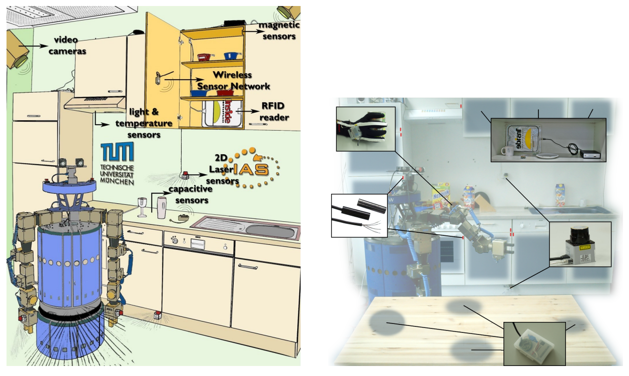
Consider, for example, an autonomous household robot that is to assist in tasks such as setting the table, preparing food or cleaning up after a meal. One of the most impressive aspects of the way in which humans deal with such problems is the way in which they handle uncertain information. When given, for instance, the task of setting the table for breakfast, people will infer who will participate in the meal, where the participants will sit, what they will eat and what utensils they will consequently require as well as where to put them. Moreover, people will adequately adjust the way in which they set the table upon receiving new information, such as that Dominik will eat cereals rather than bacon and eggs. People also have excellent heuristics on where to look for things to be put on the table, and these heuristics typically take vast amounts of context information such as recent activity in the kitchen and spatial relationships into account so as to determine a suitable order in which places should be searched. All the control decisions related to such tasks require reasoning in the light of uncertainty, and for autonomous household robots to achieve a similar level of competence in dealing with such tasks, it is vital that they be equipped with knowledge representation systems that fully support reasoning at appropriate levels of abstraction.

In particular, we want our models to be as general as possible — not specific to a particular instance of an environment, such as a certain kitchen or household. For instance, we might want to state that *anybody* who intends to eat cereals is likely to use a bowl and a tablespoon (but might also use a cup or a teaspoon). Therefore, first-order languages, which allow universal quantification and thus abstract away from concrete objects, are a suitable basis for our models. They essentially provide us with the much-needed expressive power of natural language. In recent years, numerous approaches that seek to combine first-order representations with the semantics of probabilistic graphical models have been proposed in the subdiscipline of artificial intelligence that has emerged as statistical relational learning [1]. This combination precisely addresses the two main issues in real-world environments: complexity and uncertainty. First-order representations are well-suited to dealing with high degrees of complexity by supporting universal rules that generalize across objects having similar properties; and probabilistic models allow for the representation of varying degrees of uncertainty.

A statistical relational template model typically contains a set of general first-order sentences that describe dependencies among atomic sentences pertaining to objects belonging to particular classes, the strength of which is quantified by probabilistic parameters. For any concrete set of objects belonging to these classes, the model can be compiled (via a template mechanism) into a ground model that represents a full-joint probability distribution over all the ground atoms that can be constructed from the model's set of logical predicates and the set of objects it was combined with. Suppose the template model contains only a single (weighted) rule $\forall p.$ *eats(p, Cereals)* $\rightarrow$ *uses(p, Bowl)*, which applies to all people p. If this model is combined with a set of concrete people, e.g. {*Dominik, Michael*}, the ground model will represent the full-joint distribution over the set of possible worlds implied by the ground atoms (boolean random variables) *eats(Dominik, Cereals)*, *uses(Dominik, Bowl)*, *eats(Michael, Cereals)* and *uses(Michael, Bowl)*, i.e. a distribution over $2^4 = 16$ possible worlds. Note that the representation of the full-joint gives an autonomous robot the flexibility of inferring arbitrary conditional probabilities, be they causal or diagnostic in nature.

Since we as system designers or knowledge engineers cannot generally quantify the degree of uncertainty that applies to a particular aspect of the domain in question, the probabilistic parameters of our models should be learnt from data. In the context of modelling the environment of an autonomous robot, the observations that have been made by the robot itself or have been gathered by a sensor-enabled environment and made accessible to the robot can be used as training data for parameter learning. We assume that the structure of the model, however, i.e. the specification of possible dependencies in the domain, is given by expert knowledge.

In this paper, we show how the powerful representational paradigm of statistical relational models can be integrated into robot control programs in such a way that they can be applied by a robot system to perform its tasks more adaptively, by parameterizing its plans and individual actions with the available context information. We thus describe an implementation of a coupling of robot control programs with statistical relational reasoners, which encompasses the selection of suitable representation languages, the collection of training data, the implementation of learning mechanisms and inference methods that meet the requirements of autonomous robot systems, and extensions of a robot planning language that support the interaction with the reasoning system as well as the interpretation of probabilistic results. As a running example, we consider the aforementioned application scenario, which we extensively investigate within the context of our research demonstrator, the Intelligent Kitchen of the CoTeSys (Cognition for Technical Systems) cluster of excellence: a household service robot (see Fig. 1) that is to take on common household tasks.



(a) (b)

Fig. 1. A B21 service robot with PowerCube arms in the Intelligent Kitchen: (a) Schematic view; (b) RFID sensors, laser range scanners and magnetic proximity sensors in the real environment

II. Applications of Statistical Relational Models

There are many situations in which an autonomous robot can potentially benefit from probabilistic reasoning capabilities. For example, many of the tasks of a service robot are usually under-specified in the sense that not all the information that is potentially relevant to successful handling of the task is provided when the command is given. In such situations, a robot can fill in the missing parts of task specifications by inferring, in a nutshell, the most likely specification and adjusting its control program appropriately. The task of setting the table is a good example of such a case: How the table is to be set depends on a number of factors, such as the type of the meal and, in particular, the participants and their individual requirements and habits. Setting the table for an unknown group of four people is an entirely different task than is setting the table for four known members of the family, about whom we know quite a bit from experience. In the former case, we should probably be conservative and provide all the things that could potentially be needed, while in the latter case, we should probably take any knowledge we may have about the seating order, consumption habits of the individuals and perhaps even preferred utensils into account. If we do not know who the people participating in the meal are going to be, it could make sense to infer who is most likely to take part given the type of the meal (e.g. breakfast), the day of the week and the current time. Moreover, social structures that may affect the participation of certain individuals can be accurately captured in a relational model: If we know, for instance, that Steve will take part and that Pete is Steve's friend, the model could indicate an increased probability for Pete's participation. Parameterizing a plan for table setting with the knowledge represented by and retrieved from a statistical relational model can thus be highly desirable, for it allows us to adapt our default plans to the concrete situation at hand, achieving highly individualized behaviour that is inspired by past experience.

Another possible application is the process of context-specific decision-making and plan selection. A robot that is observing humans as they take actions or is given initial commands to carry out certain actions can conclude from context information what the most likely intentions of the people it interacts with are, allowing it to anticipate necessary future actions, which can subsequently be carried out without the need for further instructions – provided that the robot has a statistical relational model that captures precisely the connection between sequences of actions and the respective context. Furthermore, the plan to select for a particular task may depend on certain facts that are not known to the robot. Inferring the most likely truth value of the corresponding facts can thus help to select appropriate plans. For instance, a robot that is supposed to deliver an object to somebody whose location or even whose presence in the household is not known can use a model of people's habits to infer the most likely location given the information that we have (e.g. time, day of the week, known facts about others) in order to select a plan that is appropriate for delivery. If the person is likely to be absent, a robot can first ascertain that the person is really absent and then take measures to ensure delivery at a later point in time or by different means.

Similarly, we can use probabilistic models to select heuristics, e.g. search heuristics. A robot that initially takes on its role as a household assistant can use models about common layouts of kitchen environments to direct its search for utensils whose locations are yet unknown by inferring the most likely locations. For instance, it might be likely for a wooden spoon to be in a drawer next to the stove if there is such a drawer. Even a robot that is already familiar with the environment may not know at all times where the objects it may require are going to be located. Given a model of how family members generally move objects around the household as events take place, the robot can, however, infer the most likely positions in order to determine a suitable search order.

III. Architectural Overview

We will now briefly describe the architecture that we implemented to support applications such as the ones described above. For the sake of modularity, the probabilistic reasoning engine and the robot controller that makes use of it are realized as separate processes that interact via remote procedure calls (RPCs). In particular, we use an RPC implementation based on the YARP platform [2], which we modified to work asynchronously.

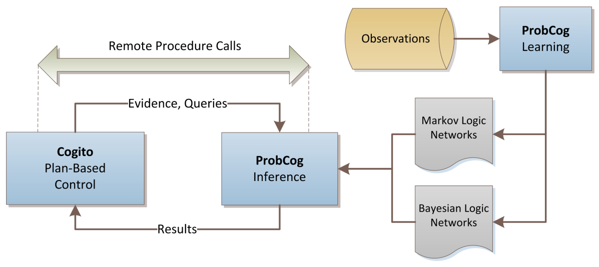


Fig. 2. Coupling of the plan-based control module (Cogito) and the probabilistic reasoning module (ProbCog)

The basic architecture is shown in Figure 2. The plan-based controller is implemented on top of an extended version of the Lisp dialect that is the reactive plan language, RPL [3]. It stores known facts about entities in the environment in a knowledge base which can be used to provide evidence to the probabilistic reasoner. Whenever the control program is faced with a situation in which probabilistic inference is necessary, e.g. an under-specified task, it queries the probabilistic reasoning system by issuing a request consisting of the name of the model to use as well as a list of evidence variables (taken from its knowledge base) and a list of query variables, where the variables are simply logical ground atoms. The ProbCog reasoner, which manages a pool of probabilistic models, then processes the request by instantiating the selected model for the given set of objects, running the inference method, and finally returning the inference results in a reply. The robot controller then processes the returned probabilities by applying suitable operators (e.g. thresholding or argmax) and uses the processed result to parameterize its plans or modify its control program in general. Section VII will describe how this is realized in more detail.

As a simple example, consider again the problem of setting the table for breakfast. Assume that in the controller's knowledge base, we have been told that exactly three people will participate, namely Anna, Bert and Dorothy — members of the family that are known to our model. To set the table, we need to know what utensils will be required at which seat; therefore if we know what utensils people will probably use and where they will sit, we have the information that we need. Our problem translates to a probabilistic query as follows,

$$\begin{aligned}&\boldsymbol{P}(\mathit{sitsAtIn}(?p, ?pl, M), \mathit{usesAnyIn}(?p, ?u, M) \mid \\ &\quad \mathit{mealT}(M, \mathit{Breakfast}) \wedge \mathit{day}(M, \mathit{Thursday}) \wedge \\ &\quad \mathit{takesPartIn}(P1, M) \wedge \mathit{name}(P1, \mathit{Anna}) \wedge \\ &\quad \mathit{takesPartIn}(P2, M) \wedge \mathit{name}(P2, \mathit{Bert}) \wedge \\ &\quad \mathit{takesPartIn}(P3, M) \wedge \mathit{name}(P3, \mathit{Dorothy}))\end{aligned} \tag{1}$$

i.e. there is a breakfast meal M, in which the three people take part, and we are interested in the probability of *sitsAtIn* atoms telling us who will sit where and *usesAnyIn* atoms telling us who will use which utensils. The query will return, for each person and place, the probability of the corresponding *sitsAtIn* atom, and, for each person and utensil type, the probability of the corresponding *usesAnyIn* atom.

IV. Representation Formalisms

In recent years, many representation formalisms that combine first-order logic or a subset thereof with probabilistic graphical models have been proposed, some based on undirected probabilistic graphical models, others on directed models. Markov logic networks (MLNs) [4] are based on the former and are among the most expressive, for they indeed support the full power of first-order logic. An MLN is essentially a first-order logic knowledge base, where the formulas correspond to soft constraints, the hardness of which is represented in probabilistic parameters attached to the formulas. Combined with a concrete set of objects/constants, an MLN defines a ground Markov random field [5] that has one boolean variable for every logical ground atom and whose features are given by ground instances of the formulas, the weights of these features being the parameters attached to the formulas. For any concrete set of constants, an MLN thus models the full-joint probability distribution over a set of possible worlds (induced by the logical ground atoms), which supports the inference of arbitrary conditional probabilities.

The expressiveness of MLNs does come at a price, however, for not only is learning generally more problematic [6], inference also becomes more expensive and is therefore less well-suited to near-real-time applications. Nevertheless, we use them in cases where the added expressiveness is key. In cases where we do not require the added expressiveness, we use a representation that is based on directed graphical models. Our representation, named Bayesian logic networks (BLNs), can more or less be regarded as a dialect of multi-entity Bayesian networks (MEBNs) [7], in which one specifies individual conditional probability distributions, MEBN fragments (MFrags), which are applicable to a random variable under certain circumstances and which collectively define a template for the construction of a Bayesian network for any given set of objects/constants. As a special feature, our language specifically supports global logical constraints on the distribution, which are realized as parameterized logical MFrags that are always part of the evidence, i.e. all instances of such constraints are required to be satisfied in all ground networks.

For added convenience, the ProbCog framework supports the conversion of BLNs to MLNs, such that learning algorithms applicable to BLNs can be used to learn MLNs and inference algorithms for MLNs can be used for BLNs. The support for such conversions also allows us to extend models with constraints unsupported by BLNs as needed, transforming them to MLNs and continuing the modelling process in the richer representation language.

V. Learning

For our models to be grounded in observations made in the real world, we support learning methods. As outlined above, we assume that the structure of the model, i.e. a specification of possible dependencies, is given by a knowledge engineer. For the table setting model, a simplified causal structure of a stochastic process that might apply to the domain is shown in Figure 3. We can adequately translate such a structure into either conditional dependencies (MFrags) or logical formulas (features of MLNs).

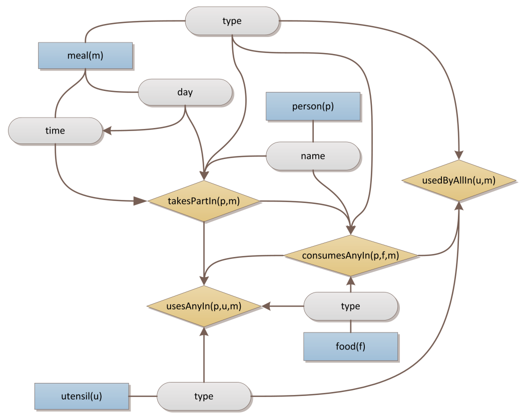


Fig. 3. Simplified structure of a stochastic process that applies to the table setting model (not all variables shown), where the arrows indicate direct (causal) influence.

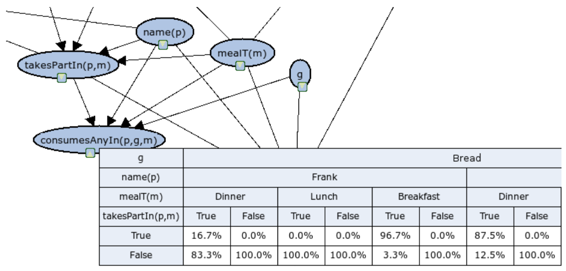
The actual ProbCog learning stage then uses a training database containing a list of ground atoms (atomic sentences that directly correspond to sensory observations) in order to learn the model parameters that most appropriately explain the observations that were made. To obtain a training database, we collect data from various sources and translate it into the logical format we require. For the purpose of data acquisition, our Intelligent Kitchen is equipped with a multitude of sensors (see Figure 1), including RFID sensors in cupboards, on tables and in gloves worn by kitchen users; laser range scanners; cameras; etc.

For the table setting model, the configurations in which the table has been set can, for instance, be observed by an overhead camera, and the point in time at which the data should be stored can be identified by defining the time-span in which a meal takes place as the time between setting the table and cleaning up. The actual generation of logical ground atoms for a set of observations is then straightforward. Consider, for example, another scenario, in which we want to record training data for a model that is concerned with pick and place actions. The following atoms representing a sequence of actions and situations can, for example, be added to a training database based on observations of RFID sensors alone:

Data	Sensor	Time	Generated atoms
ID_Cup$_3$	RFID:Cupboard$_1$	t	
ID_Cup$_3$	RFID:Glove$_{P_1}$	t+x	performed(P$_1$, A$_1$, S$_1$)
			actionT(A$_1$, Pickup)
			place(A$_1$, Cupboard$_1$)
			involves(A$_1$, Cup$_3$)
ID_Cup$_3$	RFID:Table	t+x+y	succ(S$_1$, S$_2$)
			performed(P$_1$, A$_2$, S$_2$)
			actionT(A$_2$, Putdown)
			place(A$_2$, Table)
			involves(A$_2$, Cup$_3$)

From first recognizing the RFID of Cup$_3$ in the cupboard and then recognizing it in Person$_1$'s glove, we can deduce that there was some situation S_1 in which the person performed a pick up action at the containing cupboard that involved Cup$_3$, and this information can be written using appropriate logical atoms that match the representation in our model. In the future, we plan to use more elaborate monitoring techniques for actions, such as markerless tracking of human motion based on multiple camera views [8], with added classifiers for action recognition.

The actual learning algorithms that yield parameters from the gathered training data are based on either maximum likelihood or MAP estimation. In MLNs, even learning needs to be done approximately, since an optimization of the likelihood of the parameters given the possible world embodied by the training data requires exact inference over the model, so one usually resorts to approximations. Even so, the learning problem as a whole is, unfortunately, ill-posed in the sense that there is not a single optimal solution, and different solutions generally imply different probability distributions depending on the size of the domain for which the MLN template is instantiated [6]. To solve this problem, our implementations of learning algorithms for MLNs allow the use of constrained optimization to impose necessary integrity conditions on the distributions. The problem can also be circumvented by learning instead in the BLN framework and then translating the model to an MLN — provided that the dependency structure can be captured by a BLN. In BLNs, which make the causal structure of the model explicit, maximum likelihood learning is particularly simple, as it essentially reduces to counting occurrences of parent-child configurations in the data. Figure 4 shows an exemplary part of an MFrag of the table setting model indicating the conditional distribution of the predicate *consumesAnyIn(person, food, meal)*.



g	Bread							
name(p)	Frank							
mealT(m)	Dinner		Lunch		Breakfast		Dinner	
takesPartIn(p,m)	True	False	True	False	True	False	True	False
True	16.7%	0.0%	0.0%	0.0%	96.7%	0.0%	87.5%	0.0%
False	83.3%	100.0%	100.0%	100.0%	3.3%	100.0%	12.5%	100.0%

Fig. 4. Parameters of a BLN

VI. Inference

We have high demands on the reasoning capabilities of our system, because if the probabilistic knowledge base is to be queried by a robot controller, it needs to produce results within short periods of time. Yet the results should be approximately correct nonetheless. Given the NP-hardness of probabilistic inference, exact inference is, unfortunately, infeasible in all but the smallest of domains. We therefore resort to approximate inference techniques, which are either based on independent sampling, Markov chain Monte-Carlo (MCMC) or loopy belief propagation.

For BLNs, the ProbCog inference module support various sampling algorithms. As long as domains lack deterministic dependencies and queries involve few evidence variables, standard methods such as likelihood weighting [9] or Gibbs sampling [10] will perform well. In the presence of unlikely evidence, it becomes increasingly important for algorithms to explicitly incorporate the evidence into the sampling procedure if acceptable convergence rates are to be reached — by, for instance, sampling backward from the evidence (e.g. backward simulation [11]) or propagating the effect of evidence variables before proceeding with forward sampling (e.g. importance sampling based on evidence-prepropagation, EPIS-BN [12]). In particular, we implemented backward simulation and variants thereof that take more context information as well as prior probabilities into account, achieving improved convergence rates. We furthermore support EPIS-BN and other state-of-the-art inference algorithms through the inclusion of the SMILE library [13].

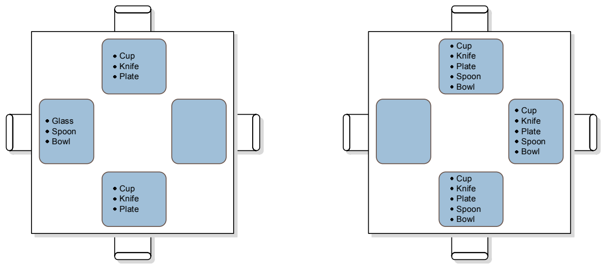
For MLNs, the only inference algorithm that has proved to produce accurate results in real-world situations is MC-SAT [14], which, unlike other algorithms, can soundly handle deterministic dependencies. Our implementation of MC-SAT supports additional constraints that are of particular importance in real-world domains, such as cardinality constraints limiting the number of objects that a given object or group of objects can be related to. (In the purely logical form, cardinality restrictions cannot be used owing to exponential blowup as a result of the normal form conversion required by the algorithm.) Moreover, we achieve improved accuracy by fully maintaining model structure, i.e. we do not decompose complex formulas into clauses. Nevertheless, the ProbCog reasoner also includes an interface to the open-source Alchemy system [15] for inference in MLNs.

Given the fact that real-world domains can be quite large, it becomes increasingly important to consider alternatives to performing inference in the ground model, i.e. the ground Markov random field or Bayesian network. Lifted inference methods, i.e. methods that explicitly exploit the repeated structure of the ground models, essentially lifting the inference problem to the first-order level, are certainly worthwhile exploring in the future [16], [17], [18].

As an example, consider the query (1). In our model, it produced results that imply the configuration shown in Figure 5(a) when assuming for each person the most likely seating location and assuming that *usesAnyIn* atoms with a probability over 0.05 should be considered to hold:

usesAnyIn(P1, Plate, M)	0.9981
usesAnyIn(P1, Fork, M)	0.0000
usesAnyIn(P1, Cup, M)	0.9136
usesAnyIn(P1, Platter, M)	0.0000
usesAnyIn(P1, Bowl, M)	0.0347
usesAnyIn(P1, Glass, M)	0.0000
usesAnyIn(P1, Knife, M)	0.9981
usesAnyIn(P1, Spoon, M)	0.0347
usesAnyIn(P1, Pitcher, M)	0.0000
usesAnyIn(P2, Plate, M)	0.9967
usesAnyIn(P2, Fork, M)	0.0000
usesAnyIn(P2, Cup, M)	0.8546
...	
sitsAtIn(P1, Seat1, M)	1.0000
sitsAtIn(P2, Seat2, M)	0.7815
...	

Notice that the results change considerably if we remove from the evidence the identities of the three people (Figure 5(b)). The video accompanying this paper shows the (simulated) robot performing specifically adapted plans based on the inference results for these two cases.



(a) (b)

Fig. 5. Interpreted inference results.

VII. Integration with the Control Program

In this section, we will describe how the probabilistic reasoning mechanism is integrated into the plan language RPL (reactive plan language) [3], which is used for controlling our kitchen robot. Plans written in RPL are the basis of a transformational planning system [19] which optimizes robot activities in a complex environment such as a human household, based on the plan source code and observations of the robot's behaviour. We therefore require our plan language to make use of highly declarative language constructs such as *with-failure-handling* (indicating failure handling and failure recovery code) or *at-location* (execution of plan steps at a specific location). RPL, like every Lisp-like language, provides a powerful mechanism to extend the language with new commands but also, in contrast to most other languages, with new special forms, e.g. for establishing variable bindings.

To integrate probabilistic inference into the planning framework, RPL is extended with a new (declarative) language construct, *likely-let*. In analogy to the Lisp special form *let*, it establishes a binding of variables to tuples of atoms and the corresponding probabilities within the current lexical context, based on a set of queries and a set of evidences. Several applications of the resulting probability distributions are conceivable. For instance, decisions may be based directly on probabilities or we may be interested in a list of the most likely atoms to parameterize a plan. Therefore, *likely-let* also provides support for post-processing returned probability distributions. When querying seating locations, we require, for each person, a single location at which to place the person's objects, which is achieved by the application of an argmax operator over the location probabilities for every person. The result of a query for utensils on the other hand should be post-processed by a

```
1   (likely-let
2     ((places
3         :query
4           '(sitsAtIn ?person ?seating-location M)
5         :argmax ?person)
6       (utensils
7         :query '(usesAnyIn ?person ?utensil M)
8         :threshold 0.05)
9       :evidence
10        '((takesPartIn P1 M) (name P1 "Anna")
11          (takesPartIn P2 M) (name P2 "Bert")
12          (takesPartIn P3 M) (name P3 "Dorothy")
13          (mealT M "Breakfast")))
14    (with-designators
15        ((table '(the entity (name kitchen-table))))
16      (for-all-matching
17          (lambda ((?person ?place ?m)
18                   (?person ?entity-type ?m))
19            (with-designators
20                ((obj (an entity (type ,entity-type)
21                                  (status unused)))
22                 (seat (a location (on ,table)
23                                    (place ,place))))
24              (achieve (entity-on-entity
25                         obj table
26                         seat))))
27          (cross-product places utensil))))
```

Fig. 6. A table setting plan that uses probabilistic inference.

threshold operator, as we want to place all the objects on the table where the usage probability is above a specific threshold.

As an example, let us consider the plan for setting the table as described in section III, where the three people named Anna, Bert and Dorothy will participate and the type of the meal is breakfast. This information is given as evidence. We query the seating locations and the objects used by the participants. The operators *argmax* and *threshold* are applied in a post processing step. In the example code shown in Figure 6, the variables *places* and *utensils* are bound to the post-processed probability distributions. The variable *places* is bound to a list containing the seating location with the highest probability for every person:

```
((P1 Seat1 M) (P2 Seat2 M) (P3 Seat0 M))
```

The variable *utensils* contains a similar but much longer list owing to the application of the threshold operator.

Iteration is performed by iterating over the matching elements of the cross product set generated by combining the elements of *places* and *utensils*. The command *for-all-matching* executes the lambda body (lines 19 – 26) only for pairs matching the pattern *((?person ?place ?m) (?person ?entity-type ?m))* (line 17).

As the example shows, probabilistic inference can provide a sound way of parameterizing under-specified plans. Moreover, as previously sketched, it can help in supporting the perception system in order to find objects more reliably in new and even unknown environments, and it can support decision-making and plan selection in general.

VIII. Conclusion

In this paper, we have outlined an architecture that equips robot control programs with first-order probabilistic reasoning capabilities. We mentioned possible applications in the context of a household service robot and we described our implementation of both the probabilistic knowledge representation component, which integrates learning and inference in two formalisms, and the actual robot controller, which modularly and transparently extends a robot plan language to include the results of probabilistic inference into the process of plan generation.

We strongly believe that any intelligent autonomous system that is to act competently in a real-world environment must accurately represent knowledge about its environment, and, given the high degree of uncertainty of real-world domains, it is clear that at least some parts of this knowledge must be probabilistic. The architecture we proposed can be seen as a first step towards a cognitive system that represents complex knowledge in a declarative, flexible way and that is in a position to employ that knowledge in ways that support adaptive and reliable behaviour.

IX. Acknowledgements

This work is supported by the CoTeSys (cognition for technical systems) cluster of excellence.

References

[1] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[2] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: Yet Another Robot Platform," *International Journal of Advanced Robotics Systems, special issue on Software Development and Integration in Robotics*, vol. 3, no. 1, 2006.

[3] D. McDermott, "A Reactive Plan Language," Yale University," Research Report YALEU/DCS/RR-864, 1991.

[4] M. Richardson and P. Domingos, "Markov Logic Networks," *Mach. Learn.*, vol. 62, no. 1-2, pp. 107–136, 2006.

[5] J. Pearl, *Markov and Bayes networks: A comparison of two graphical representations of probabilistic knowledge (CSD. University of California at Los Angeles. Computer Science Department)*. University of California at Los Angeles, 1986.

[6] D. Jain, B. Kirchlechner, and M. Beetz, "Extending Markov Logic to Model Probability Distributions in Relational Domains," in *Proceedings of the 30th German Conference on Artificial Intelligence (KI-2007)*, 2007, pp. 129–143.

[7] K. B. Laskey, "MEBN: A language for first-order Bayesian knowledge bases." *Artif. Intell.*, vol. 172, no. 2-3, pp. 140–178, 2008.

[8] J. Bandouch, F. Engstler, and M. Beetz, "Accurate human motion capture using an ergonomics-based anthropometric human model," in *Proceedings of the Fifth International Conference on Articulated Motion and Deformable Objects (AMDO)*, 2008.

[9] R. M. Fung and K.-C. Chang, "Weighing and Integrating Evidence for Stochastic Simulation in Bayesian Networks," in *UAI*, 1989, pp. 209–220.

[10] W. R. Gilks, *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC, December 1995.

[11] R. M. Fung and B. D. Favero, "Backward Simulation in Bayesian Networks," in *UAI*, 1994, pp. 227–234.

[12] C. Yuan and M. J. Druzdzel, "An importance sampling algorithm based on evidence pre-propagation," in *UAI*, 2003, pp. 624–631.

[13] M. J. Druzdzel, "SMILE: Structural Modeling, Inference, and Learning Engine and GeNIE: A Development Environment for Graphical Decision-Theoretic Models," in *AAAI/IAAI*, 1999, pp. 902–903.

[14] H. Poon and P. Domingos, "Sound and Efficient Inference with Probabilistic and Deterministic Dependencies." in *AAAI*. AAAI Press, 2006.

[15] S. Kok, P. Singla, M. Richardson, and P. Domingos, "The Alchemy system for statistical relational AI," http://alchemy.cs.washington.edu/, 2004.

[16] D. Poole, "First-order probabilistic inference," in *IJCAI*, 2003, pp. 985–991.

[17] R. de Salvo Braz, E. Amir, and D. Roth, "Lifted First-Order Probabilistic Inference," in *IJCAI*, 2005, pp. 1319–1325.

[18] P. Singla and P. Domingos, "Lifted first-order belief propagation," in *AAAI*, 2008, To appear.

[19] A. Müller, "Transformational planning for autonomous household robots using libraries of robust and flexible plans," Ph.D. dissertation, Technische Universität München, 2008.