Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities

Nicholas M. Stiffler Ashok Kumar

April 2011

Outline

Abstract

Introduction

Applications of S.R. Models

Architectural Overview

Representation Formalisms

Learning

Inference

Integration with the Control Program

References

Abstract

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.

Introduction

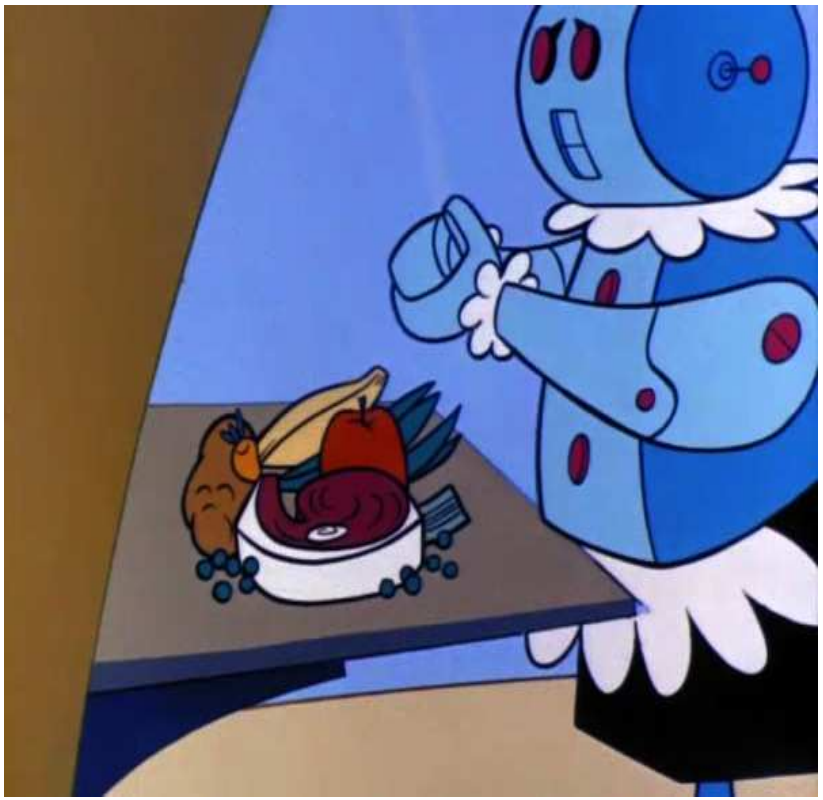
Problem Introduction

Autonomous Robot in human environments

Why the is there a need for powerful Probabilistic Reasoning?

- To make reasonable predictions about the state of the environment.
- High dimensional State Estimation problems.
- Robust control programs
  (i.e anticipate failures, select actions that maximize success likelihood).
- Allows the robot to extract *MEANING* behind agent interaction.

Example



Setting the breakfast table – Uncertain Info

– Who's eating, who sits where, who eats what, who's going to need Utensil x, y, z.

Incorporating Reasoning

The Human way –

Knowledge that Person X always eats cereal.

Subconscious Heuristics for locating plates/bowls and utensils.

contextual info –

- ⋆ recent activity in the kitchen
  (has someone recently run the dishwasher)
- ⋆ spatial relationships

All of this information is taken into account when deciding the order in which places should be searched.

All of these decisions require reasoning under uncertainty, how can this be adapted so that a Robot can solve similar tasks.

Robot Knowledge Representation

What Does the Robot Need?

A Knowledge Representation that supports reasoning at appropriate levels of abstraction.

- The model needs to be as general as possible, and not specific to a particular instance of an environment.

First-order Languages, allow universal quantification, and abstract away concrete objects.

Anybody who eats cereal is likely to use a bowl and a tablespoon. But may use a cup or teaspoon.

Statistical Relational Learning

SRL is a combination of:

- First Order Representations
- the semantics of probabilistic graphical models.

Q. How does this affect Real-World environments?

A. 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.

Probabilistic Models –

Allow for representing the varying degrees of uncertainty.

Statistical Relational Template

- 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 the classes, the model can be compiled(via 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.

Example Template Model

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.$\{$Nick,Ashok$\}$.

The ground model will represent the full-joint distribution over the set of possible worlds implied by the ground atoms(*boolean r.v.'s*)

$$eats(Nick, Cereal), uses(Nick, Bowl),$$
$$eats(Ashok, Cereal), uses(Ashok, Bowl)$$

a distribution over $2^4 = 16$ possible worlds.

Probabilistic Parameters

Generally it is not possible to quantify the degree of uncertainty that applies to a particular aspect of the domain in question.

So the probabilistic parameters of the models should be learnt from data.

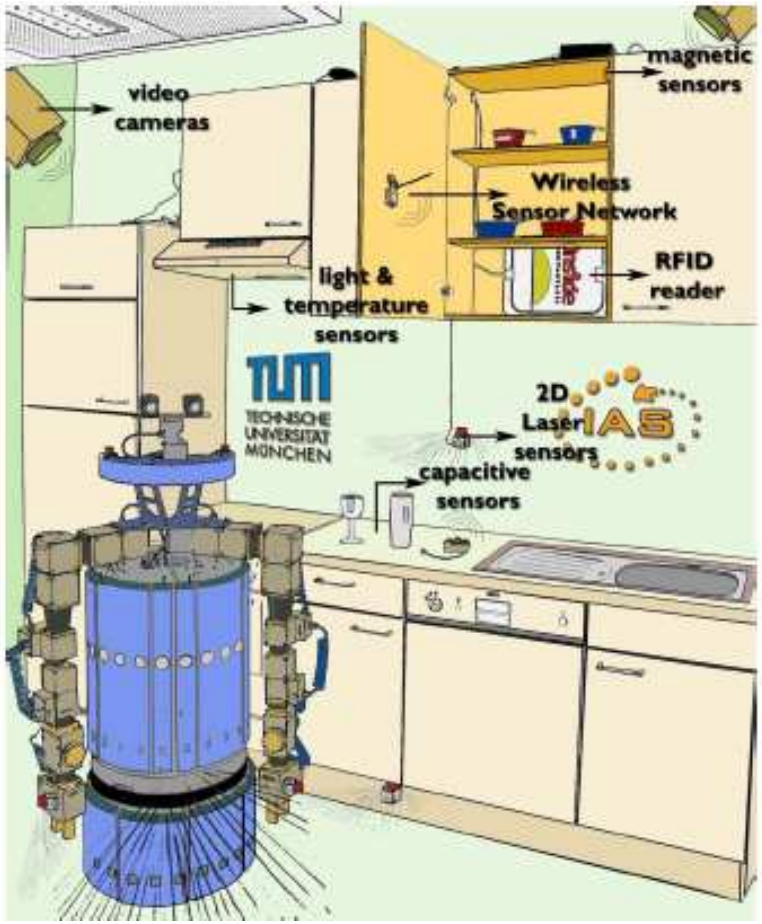
(Training data for parameter learning)

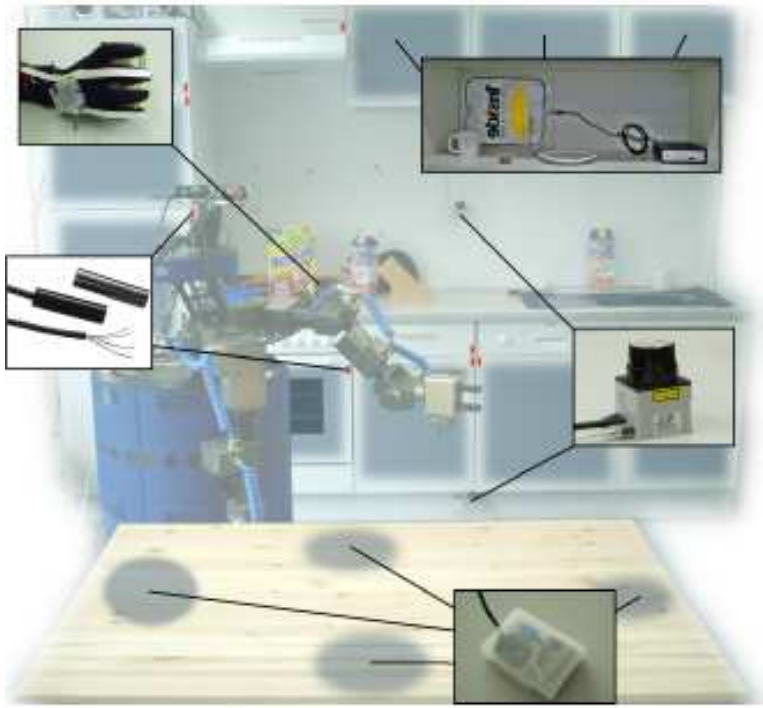
Note: The structure of the model

(spec. of possible dependencies in the domain)

is given by expert knowledge.

Pictures





Applications of S.R. Models

Benefits from Probabilistic Reasoning Capabilities

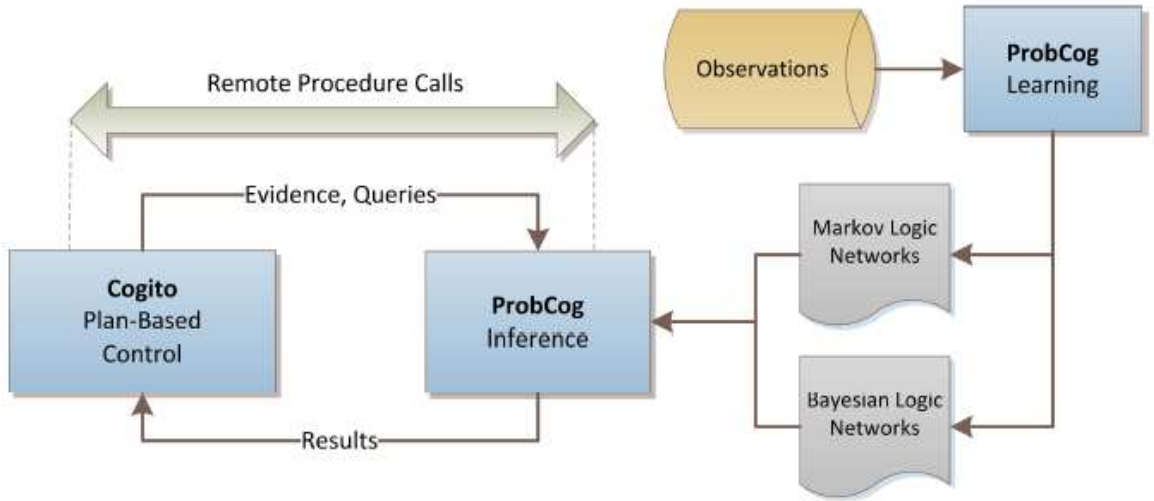
- Tasks are usually under-specified.
  - A Robot can fill in the missing parts of a task specification by inferring the most likely specification and adjusting its control program appropriately.
  - Setting the table.
  - Parametrizing a plan with knowledge represented by an retrieved from a statistical relation model is desirable in that it allows the adaptation of default plans to a concrete situation.
- Context Specific Decision-Making and Plan Selection
  - Given a statistical relational model that captures precisely the connection between sequences of actions and the respective contexts.
  - In the absence of facts, infer a ”logical”(most likely) truth value to create an appropriate plan.

Benefits from Probabilistic Reasoning Capabilities Cont.

- Use probabilistic models to select heuristics
  - Search Heuristic (for an item that is unknown) – Infer from common kitchen layouts, the most likely location of a utensil.
  - Even known items can rely on a search
    Ex. Is what I'm looking for in the dishwasher
    Has . . . moved the item to another spot.

Architectural Overview

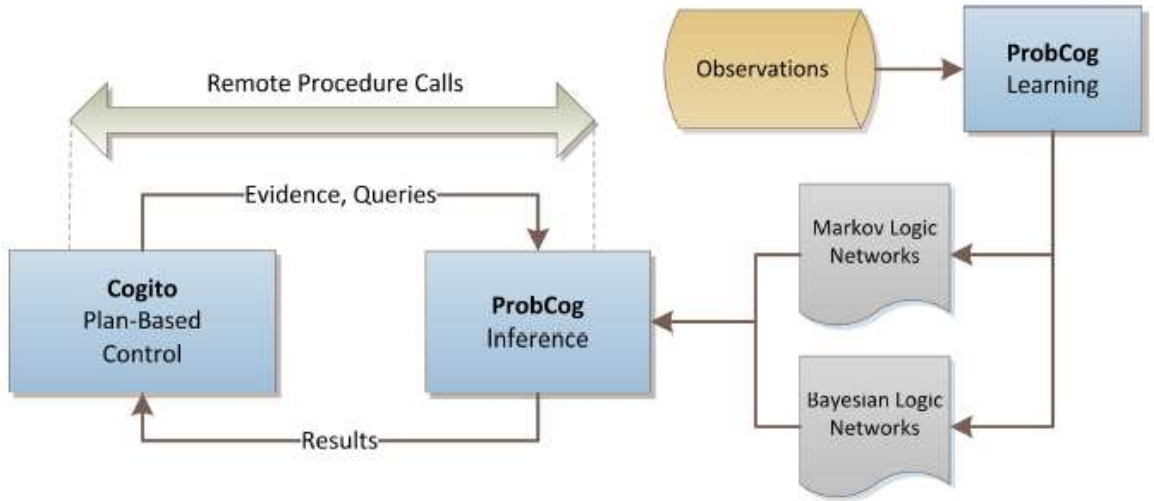
The Architecture



Probabilistic Reasoning Engine interacts with the Robot Controller using RPC's. Plan Based Controller is implemented on top of an extended version of the Lisp dialect, RPL.

- It stores known facts about entities in the environment in a KB which can be used to provide evidence to the probabilistic reasoner.

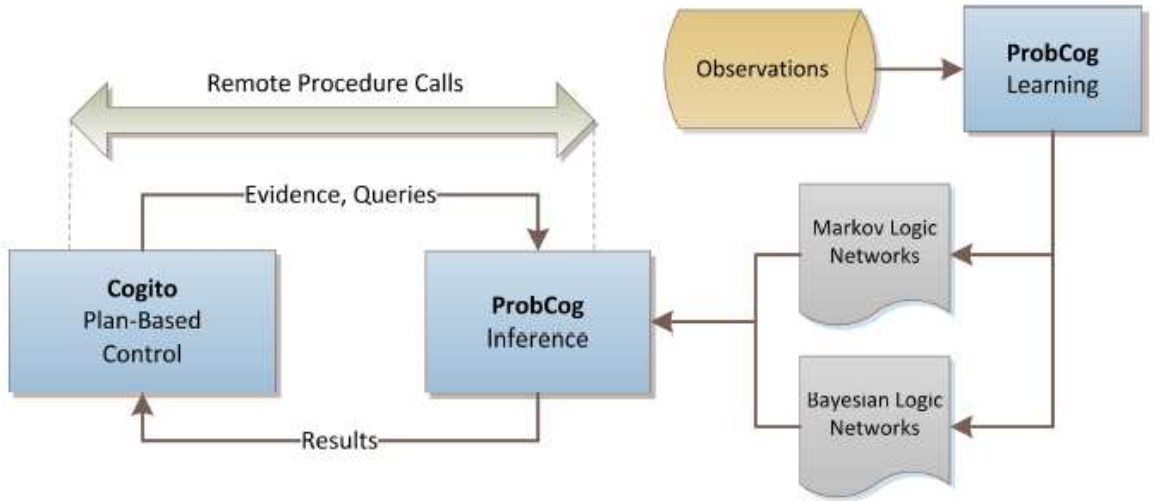
The Architecture Cont.



When the Control Program is faced with a situation in which probabilistic inference is necessary e.g. an under-specified task it query's the p.r. system by issuing a request consisting of:

- the name of the model to use
- a list of evidence variables(taken from KB)
- a list of query variables (variables are logical ground atoms)

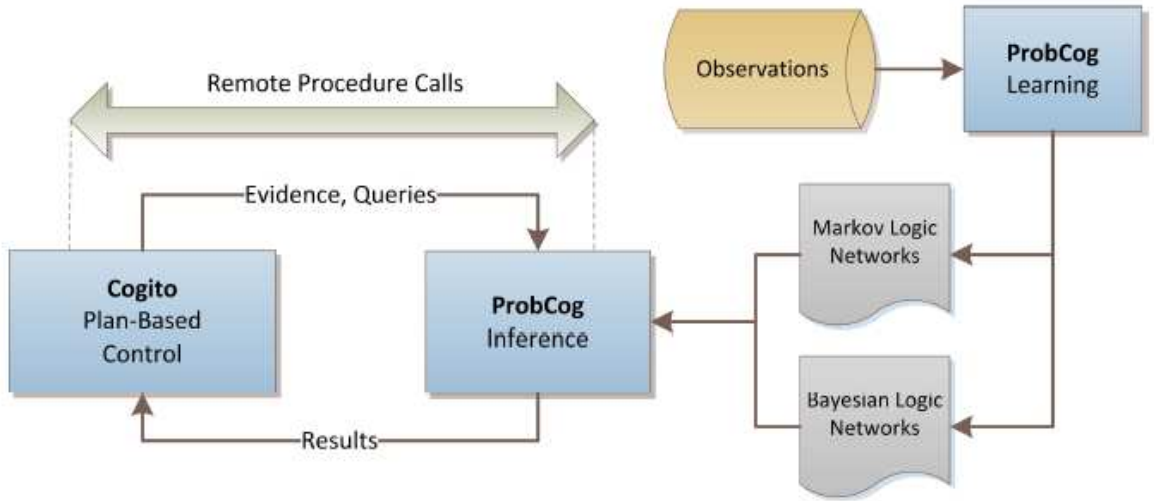
The Architecture Cont.



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
- returning the inference results in a reply.

The Architecture Cont.



The Robot Controller then processes the returned probabilities by applying suitable operators:

- thresholding, or
- argmax

and uses the processed result to parametrize its plans or modify its control program in general.

Example

Problem: Setting the table.

In KB: 3 people will be participating $\{$Moo Moo, Piggy, and Blue$\}$

These are members of Nick's family that are known to the model.

Goal: To set the table the following info is needed:

- what utensils to put at which seat.

If info regarding what utensils people will probably use, and their seating order, then the robot has the information that it needs.

This translates to the following probabilistic query:

$$P(sitsAtIn(?p, ?pl, M), usesAnyIn(?p, ?u, M) \mid$$
$$mealT(M, Breakfast) \wedge day(M, Saturday) \wedge$$
$$takesPartIn(P1, M) \wedge name(P1, MooMoo) \wedge$$
$$takesPartIn(P2, M) \wedge name(P2, Piggy) \wedge$$
$$takesPartIn(P3, M) \wedge name(P3, Blue))$$

Representation Formalisms

Markov Logic Networks

A representation formalism that combines first-order logic with undirected probabilistic graphical models.

Logical Language – FOL

Probabilistic Language: – Markov Networks

Syntax: FO formulas with weights

Semantics: Templates for Markov net features

Learning:

Parameters: Generative or discriminative

Structure: Inductive Logic Programming with arbitrary clauses and MAP score

Inference:

Maximum a Posteriori Probability (MAP): Weighted satisfiability

Marginal: MCMC with moves proposed by SAT solver

Partial grounding + lazy inference

FROM: Pedro Dominigos Dept of Computer Science & Engr.
University of Washington.

Downfalls

The expressiveness of MLN's does come at a price.

- learning is generally more problematic
- inference becomes more expensive and is therefore less well suited to near real-time applications.

Despite these drawbacks. MLN's are useful when expressiveness is key.

When the added expressiveness is not needed. They use a representation based on directed graphical models.

Use BLN's(Bayesian Logic Networks).

Bayesian Logic Networks

ProbCog

The *ProbCog* framework supports the conversion of BLNs to MLNs. This means that:

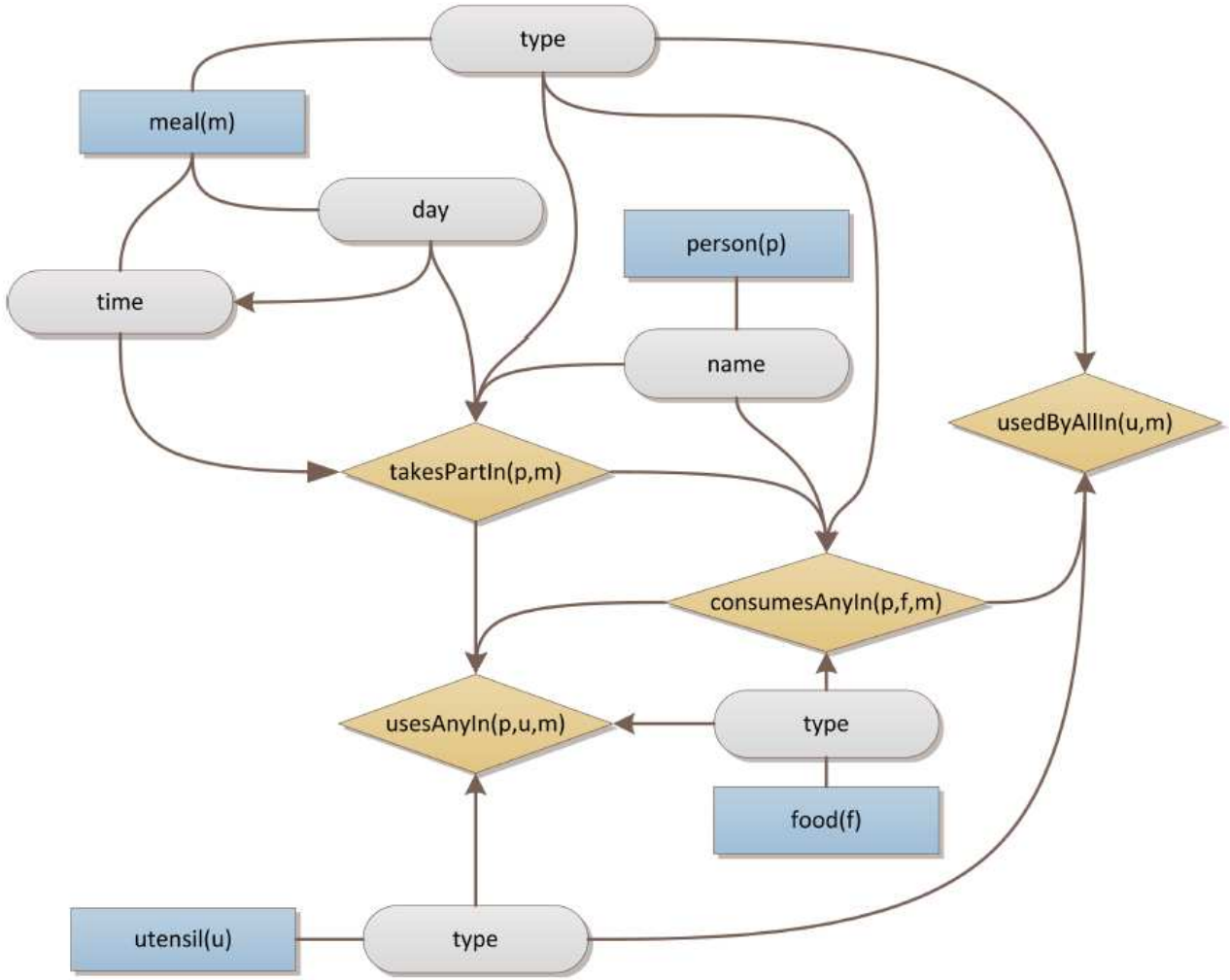
- Learning algorithms applicable to BLNs can be used to learn MLNs, and
- Inference algorithms for MLNs can be used for BLNs.

The support for conversions allows the extension of models with constraints unsupported by BLNs as needed, transforming them to MLNs and continuing the modelling process in the richer representation language.

Learning

Overview

Note: The structure of the model(specification of possible dependencies), is given by a knowledge engineer.



Explanation

The structure on the previous slide can be translated into either:

conditional dependencies – MFrags

logical formulas – features of MLNs

The ProbCog learning stage 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.

Q. How to obtain a training database?

A. Use sensors.

Example

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	$succ(S_1, S_2)$ $performed(P_1, A_2, S_2)$ $actionT(A_2, Putdown)$ $place(A_2, Table)$ $involves(A_2, Cup_3)$

Actual Learning

The learning algorithms that yield parameters from the gathered training data are based on either ML or MAP estimation.

In MLNs learning needs to be done approximately.

- The learning problem itself is ill-posed in the sense that there is not a single optimal solution.

Q. How do the author's tackle this problem?

Their implementations of learning algorithms for MLNs allow the use of constrained optimization to impose necessary integrity conditions on the distributions.

Solution $\#2$.

Solution #2

The problem can also be circumvented by learning in the BLN framework and then translating the model to a MLN.

- provided that the dependency structure can be captured by the BLN.

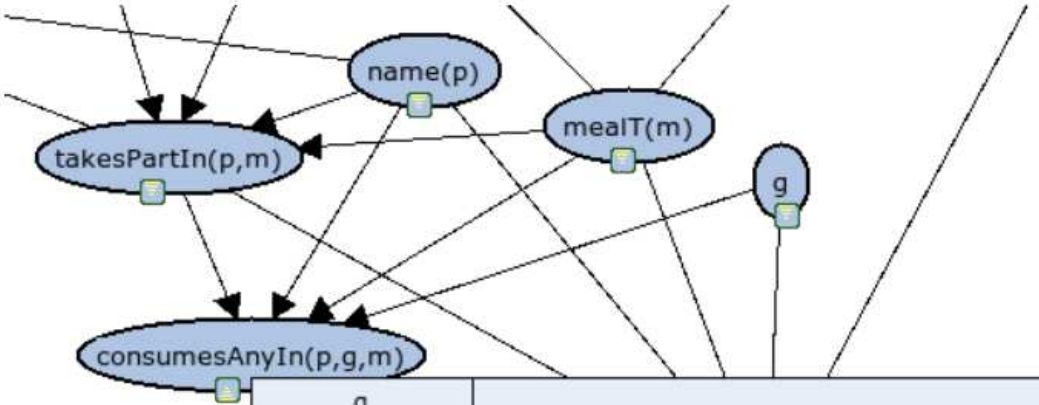
Q. Why does this work.

BLNs make the causal structure explicit, which makes ML easier because it reduces to counting occurrences of parent-child configurations in the data.

Figure

This is a part of an MFrag of the table setting model indicating the condition 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%

Inference

Hardness

There are high demands on the reasoning capabilities of the system. (Real-time).

Probabilistic Inference is NP-Hard, and exact inference is ”realistically” infeasible.

So the authors resort to the following approximate inference techniques:

independent sampling

MCMC – Markov chain Monte Carlo

> A class of algorithms for sampling from probability distributions based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. The state of the chain after a large number of steps is then used as a sample from the desired distribution. The quality of the sample improves as the number of steps increases.

loopy belief propagation

Inference for BLNs

The ProbCog supports various sampling algorithms.

As long as domains lack deterministic dependencies and queries involve few evidence variables, standard methods such as:

- likelihood weighting
- Gibbs sampling(MCMC algo)

In the presence of unlikely evidence, its important to explicitly incorporate the evidence into the sampling procedure if acceptable convergence rates are to be reached:

sampling backward from the evidence – backward simulation propagating the effect of evidence variables before proceeding with forward sampling –

- importance sampling based on evidence-prepropagation
- EPIS-BN

Inference for MLNs

The only inference algorithm with acceptable results is:

MC-SAT – combines ideas from MCMC and satisfiability.

Is based on Markov logic, which defines Markov networks using weighted clauses in FOL.

From MCMC – MC-SAT – is a slice sampler with an auxiliary variable per clause, and with a satisfiability-based method for sampling the original variables given the auxiliary ones.

From SAT – MC-SAT – wraps a procedure around the SampleSAT uniform sampler that enables it to sample from highly non-uniform distributions over satisfying assignments.

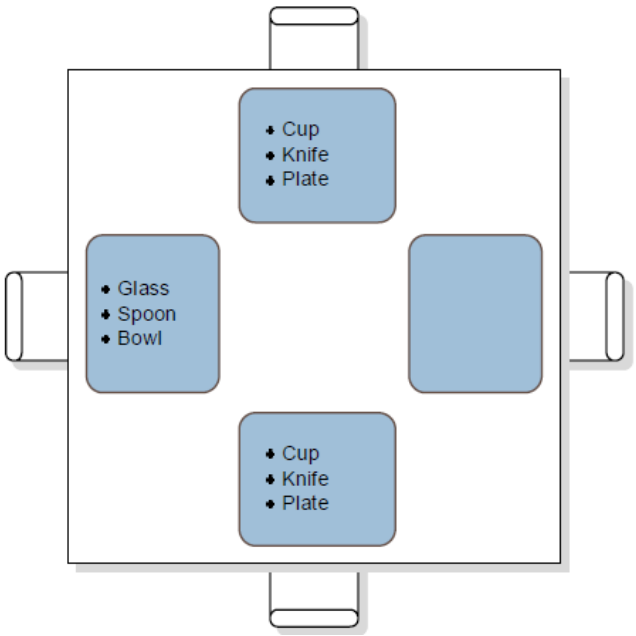
Author Adaptations

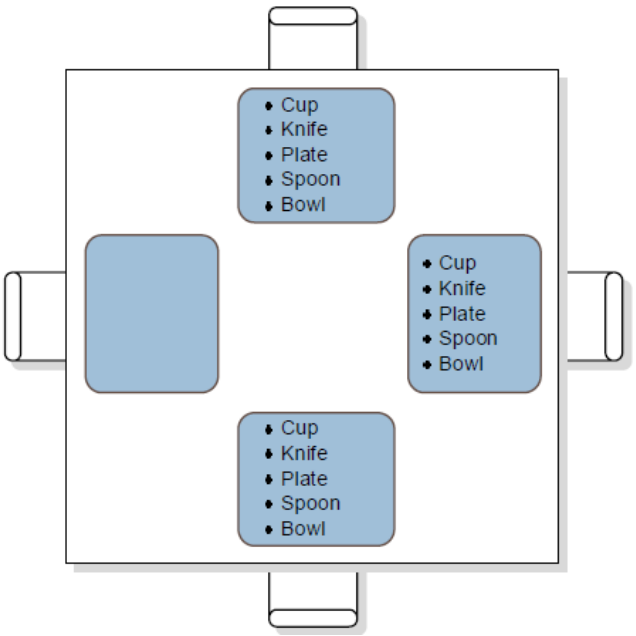
- cardinality constraints –
  (the number of objects that a object/group can be related to)
- fully maintaining model structure –
  do not decompose complex formulas into clauses.

Figure

Consider the query:

$$P(sitsAtIn(?p, ?pl, M), usesAnyIn(?p, ?u, M) \mid mealT(M, Breakfast) \wedge day(M, Saturday) \wedge takesPartIn(P1, M) \wedge name(P1, MooMoo) \wedge takesPartIn(P2, M) \wedge name(P2, Piggy) \wedge takesPartIn(P3, M) \wedge name(P3, Blue))$$





Integration with the Control Program

Integration

How is the Probabilistic Reasoning system integrated into the *plan language*(RPL), which is used for controlling the kitchen robot.

Plans written in RPL are the basis of a transformational planning system.

The plan language needs to make use of highly declarative language constructs such as:

with-failure-handling – indicating failure handling and failure recovering code.

at-location – execution of plan steps at a specific location.

Note: RPL like every Lisp-like language, provides a powerful mechanism to extend the language with new commands.

Note: RPL also allows the addition of new special forms (e.g. for establishing variable bindings)

How the Probabilistic Inference is Integrated

RPL is extended with a new declarative language construct *likely-yet*.

> It's analogous to the Lisp special from let, establishing 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:

- decisions may be based directly on probabilities
- or the user may be interested in a a list of the most likely atoms to parametrize a plan.

How the Probabilistic Inference is Integrated Cont.

Likely-yet provides support for post-processing returned probability distributions.

When querying seat locations: argmax

Utensils should be post-processed by a threshold operator

Example


```
(likely-let
  ((places
     :query
      '(sitsAtIn ?person ?seating-location M)
     :argmax ?person)
   (utensils
     :query '(usesAnyIn ?person ?utensil M)
     :threshold 0.05)
   :evidence
     '((takesPartIn P1 M) (name P1 "Anna")
       (takesPartIn P2 M) (name P2 "Bert")
       (takesPartIn P3 M) (name P3 "Dorothy")
       (mealT M "Breakfast")))
  (with-designators
    ((table '(the entity (name kitchen-table))))
   (for-all-matching
     (lambda ((?person ?place ?m)
              (?person ?entity-type ?m))
       (with-designators
         ((obj (an entity (type ,entity-type)
                          (status unused)))
          (seat (a location (on ,table)
                            (place ,place))))
         (achieve (entity-on-entity
                    obj table
                    seat))))
     (cross-product places utensil))))
```

References

Main Reference

Equipping Robot Control Programs with First-Order Probabilistic Reasoning Capabilities

Authors: Dominik Jain, Lorenz Mosenlechner, Michael Beetz

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.

Conference: International Conference on Robotics and Automation - ICRA , pp. 3626-3631, 2009.

Supplementary

Pedro Dominigos
Dept of Computer Science & Engr.
University of Washington

mlns-april-28.odp