EU FP7 CogX
ICT-215181
May 1 2008 (52 months)

DR 1.2:
Unifying representations of beliefs about beliefs and knowledge producing actions

Jeremy Wyatt, Geert-Jan Kruijff, Pierre Lison, Michael Zillich, Thomas Mörwald, Kai Zhou, Michael Brenner, Charles Gretton, Patric Jensfelt, Kristoffer Sjöö, Andzrej Pronobis, Matej Kristan, Marko Mahnič, Danijel Skočaj

⟨cogx@cs.bham.ac.uk⟩

Due date of deliverable:	March 31 2010
Actual submission date:	March 31 2010
Lead partner:	BHAM
Revision:	v2
Dissemination level:	PU

Representing the epistemic state of the robot and how that epistemic state changes under action is one of the key tasks in CogX. In this report we describe progress on this in the first period of the project, and set out a typology of the representations of epistemic knowledge we use in the project. We describe the specific representations we have developed for different domains or modalities, or are planning to develop, and how those are related to one another.

Executive Summary

This report presents the work during the first 21 months in CogX on representations of belief, of gaps and uncertainty. It also describes how those representations have been used to build systems to test our approach during the first reporting period. The task addressed was Task 1.1, of which we said

> Task 1.1 **Beliefs and beliefs about knowledge producing actions.** We will examine how a system can represent, in a unified way, beliefs about incompleteness and uncertainty in knowledge. This will start with work on their representation that will feed into WPs 2, 3 & 4, and it will later unify the modality specific representations of incompleteness and uncertainty coming up from these packages. Representations of knowledge producing actions will utilise these to represent the preconditions and effects of knowledge producing actions. These knowledge action effects will be used in WP4 for planning information gathering and processing. This task will also support work on introspection.

Here we gather together in one place technical summaries of the representations employed in different modalities, and describe the framework we are exploring at the moment for unifying those at a high level (Markov Logic). This approach augments our previous approach to binding my making it probabilistic, and thus enabling us to integrate uncertain evidence through the binder in a principled manner. We also relate them through a initial typology of knowledge gaps and uncertainties.

Role of representations of beliefs in CogX

Representations of what an agent does and doesn't know are central to CogX, so it is important for the consortium that we reflect on the relationships between these representations in different modalities. It is essential in CogX that we employ different specialised representations for different modalities and tasks. The challenge is how to unify these into abstract logical representations useful for planning and goal management. The approach in CogX is representationally driven.

Contribution to the CogX scenarios and prototypes

In this report Annex 1 presents a journal paper that describes the overall approach of CogX, the representations of gaps developed in the first year of the project, and the first year Dora and George systems. The representations

developed in Annex 2, and in the main body of the report are necessary for the year 2 scenarios involving George, Dora and Dexter. The Markov Logic approach to binding will be used in Dora and George, and various of the modality specific representations will be employed in different demonstrators for each of the scenarios.

1 Tasks, objectives, results

1.1 Planned work

Work reported in this deliverable mainly concerns Task 1.1:

> Beliefs and beliefs about knowledge producing actions. We will examine how a system can represent, in a unified way, beliefs about incompleteness and uncertainty in knowledge. This will start with work on their representation that will feed into WPs 2, 3 & 4, and it will later unify the modality specific representations of incompleteness and uncertainty coming up from these packages. Representations of knowledge producing actions will utilise these to represent the preconditions and effects of knowledge producing actions. These knowledge action effects will be used in WP4 for planning information gathering and processing. This task will also support work on introspection.

1.2 Actual work performed

We have developed a large number of representations of what the agent does and doesn't know. In the demonstrators Dora and George we employed these to drive learning. We describe these different representations and how there were used together in the overall architecture. In particular this describes our approach to dealing with multiple representational formalisms in a single system. Briefly put we produce abstractions of processing results within each sub-architecture, and then convert these abstractions into logical representations of the environmental state. This is then used to support activities such as high-level activity planning and dialogue. A paper describing this has been submitted to a special issue of IEEE Transactions on Autonomous Mental Development on Architectures and Representations. In addition we have now developed a probabilistic approach to the binding problem using Markov Logic that is described in a paper submitted to IEEE RO-MAN. We will be using this in future integrated systems. Both these papers are included as annexes to this deliverable. Finally we have produced a report which brings together the various representations of gaps and uncertainty, and places them in a typology.

1.3 Relation to the state-of-the-art

In this report we describe representations for vision, spatial representation, dialogue, planning and binding that capture uncertainty and gaps. In some areas, such as vision, our approach is an extension of approaches such as Bayesian recursive filtering, and our contribution has been in developing representations that support incremental refinement of the space of hypotheses

of what is in the visual scene. In our work on spatial representations we develop a layered, hierachical spatial model that represents space at different levels of abstractions and with different levels of granularity. The spatial representation is designed with more than navigation in mind and supports higher level tasks such as reasoning, planning and communication with humans. One of the design principles is that one should not use more details and accuracy than what is needed. In this report we describe our design, and show how gaps in the model are captured. While we employ low level representations similar to traditional SLAM [36] we also represent qualitative gaps in higher level aspects of the map, such the conceptual map, or in the locations of objects in rooms. This explicit representation of gaps at a number of different levels in the map is new. In dialogue we develop rich belief models of the situated context that capture not only what the agent believes privately, but also what other agents believe and what beliefs are shared. Thus the epistemic state of multiple agents is captured. In planning we have extended PDDL representations in several ways. We describe our decision theoretic extension of PDDL, called DTPDDL, and also how we represent epistemic effects of actions in continual planning. These representations build on our previous work in projects like CoSy, but with DTPDDL we now have basis for representing uncertainty quantitatively in domain descriptions. In binding we move beyond our previous work by employing Markov Logic rather than feature comparisons, which is a well established formalism for uncertain reasoning. Finally we have implemented these all in systems that reason about their own epistemic state. Dora, in particular plans how to achieve epistemic goals she sets herself. The work most closely related to this is that of Kaelbling and Rosenschein on the Flakey project, and the KWIK framework of Littman and co-workers. Dora goes beyond these in that she employs a much wider range of representations, and is able to synthesise plans on the fly that involve much more varied and complex representations.

2 Introduction

Central to the approach in CogX is the notion of *self-understanding*. We define this as an agent being in possession of representations and algorithms that explicitly represent and reason about what that agent does and doesn't know. This report gathers together the different types of representations we employ in the project, and relates them together through a typology. This is a first, but important step to providing a unified framework for representations that support self-understanding. In the project we have already written often about *gaps* and *uncertainty* in knowledge. These are not the same, but what useful definitions of them can we arrive at, and what different types of gaps and uncertainty are there? To help us, while it is not an entirely satisfactory term, we use *incompleteness* as an umbrella term to cover many different types of *knowledge gaps* and *uncertainty about knowledge*. Broadly we think about *uncertainty* as being concerned with non-determinism within an essentially closed world, whereas a *gap* we think about as being concerned with an open world, where the hypotheses can include previously unexperienced situations. From now on in the introduction to the typology we will avoid using general terms like *gap* and *uncertainty* and replace them with more specific terms. But as a simple example suppose a human asks the robot the category of an object. There may be a number of possible *hypotheses*, i.e. ones that are not entirely inconsistent with the observed data. Perhaps the robot sees a cylinder like shape, and it has categories for solid cylinder, pen and mug. In this case there is uncertainty about which hypothesis is correct. Alternatively suppose the robot sees a new shape which isn't consistent with any known object category. In this case there is a gap, the robot doesn't know the category and knows that it is not any of the pre-existing categories.

Having defined these basic ideas we can think about a typology of incompleteness in knowledge based on several dimensions of variability. These are *the nature of the incompleteness*, the *type of knowledge that is incomplete*, the *scope of the knowledge's applicability in time and space*, the *scope of the knowledge's applicability with respect to multiple agents* (i.e. whether the agent models its own beliefs, those of another agent, or beliefs shared between it and others), and finally whether the incompleteness is represented in a *quantitative or qualitative* manner. We illustrate these, together with some examples for selected points in the typology (Figure 2). Clearly the space of all possible points in our typology is too large to cover in a single report, and we have far from exhausted the space here. The typology itself will need adjusting over time.

With regard to the nature of the incompleteness, in the simplest case we may have a variable or variables that have a defined set of possible values or hypotheses from which the true value is known to be drawn. We refer to this as *variable value uncertainty*. We can also have uncertainty about

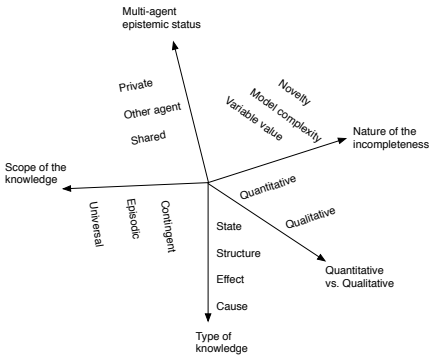


Figure 1: Dimensions of variation in gaps and uncertainty.

the number of variables needed in a model, i.e. about the *model complexity*. Finally we can also have cases where the agent knows that a variable is of an unexperienced class, i.e. that it is experiencing *novelty*. This can include cases where the variables are continuous but where the observation models for a class are quite confident and do not generalise well to some new observation. The type of knowledge that is incomplete may vary enormously. Four simple types that cover a variety of cases include contingent knowledge about the current world *state*, *structural* knowledge about the relationships that typically hold between variables, knowledge consisting of *predictions* of action outcomes or events, and knowledge about their *causes*. Finally there is a question about whether the representation is qualitative or quantitative. In qualitative representations of gaps or uncertainty we have a set of possible values for the variable, or it is simply known that the variable value is unknown. In quantitative representations we will have probabilities attached to hypotheses. Note that by a *quantitative gap* or *quantitative uncertainty* we do not mean that the underlying space for the variable is continuous or discrete, but instead that the way the incompleteness is represented involves an expression of preference for one hypothesis versus another.

Based on these distinctions we can give some examples of specific types of uncertainty and gaps that populate the space defined by the typology. In the remaining sections we will relate the representations we have developed to these. It is very important to note that this set of examples and indeed the whole typology is necessarily incomplete, and will be refined as we understand better the representations employed.

1. State variable value uncertainty: here a possible set of values for a variable describing part of the environment state (e.g. the identity of an object) is known. But the specific value that holds for a specific situation is unknown. If the uncertainty is qualitatively represented the set will be all the information the agent has. If the uncertainty is

quantitatively represented it will be the case that a probability density is defined over that set.

2. Uncertainty about state model complexity: here it is uncertain how many features there are of a particular type (e.g. how many objects there are in the room). Again if the uncertainty is qualitative it will the possible numbers of objects will be known. If it is quantitatively captured there will be a probability distribution over that set. Other examples include not knowing how many possible rooms there are in building, or how many categories of rooms are possible, or how many different equivalent configurations two objects might have.

3. State novelty: it is possible that the value of a variable is not drawn from the set of normal experienced values for that variable type, e.g. this is a colour of object I haven't seen before. It could be that there is some combination of this with *state value uncertainty*. In this case there may be a likelihood that the variable value is novel or not. In the case of continuous variables this would be a value that lies outside the previously existing range of values.

4. Structural variable value: structural knowledge defines how the variables in an environmental model, i.e. a model of state, are related to one another. Examples of this include the relationship between two variables in an ontology. For example a kitchen is a sub-type of room, and a particular kitchen is an instance of that type. Kitchens contain objects such as mugs, cookers and sinks. Alternatively having an association between variables is also a type of structural knowledge. There is for example a particular subset of the hue space that is conventionally labelled blue by most English speakers, or it may be known that one location is directly linked to another. All these structures may have uncertainty as to whether relationships exist within a set of variables, and if so what those relationships might be.

5. Structural model complexity: for some kinds of structural knowledge it may be useful to express the uncertainty about the possible structural complexity of an agent's models. In map learning, for example there may be some uncertainty about how many places in a building are directly connected to one another.

6. Structural novelty: it may be the case that an ontology does not capture the current type of experience adequately, and that new types need to be added to the ontology. So that for example, there may be no notion that there is a kind of thing called a colour, but that after learning an associative model of blue, red, green and yellow the learner becomes aware that these labels all refer to portions of a similar space. It may be that if the wrong representation is used, e.g. RGB is used

to encode colour, that the space must be re-represented in order to separate one kind of variation from another, e.g. the brightness of a colour from the hue. Spotting structural novelty means spotting the gap in an ontology, or spotting that the relationships between variables is new.

7. Effect value uncertainty: this concerns cases where the effect of an action is not determined, but drawn from a known set. Possibly the likelihood of particular outcomes may be known. Later on we will capture this sort of effect value uncertainty using DTPDDL.

8. Effect model complexity: it may simply be uncertain as to how many effects of an action there are. We do not handle this case explicitly in our current representations, but we can conceive of extensions to them that can hypothesise explicitly that there may be more outcomes than already observed. This is handled in stochastic processes for example by employing a hierarchical prior that gives a distribution over the number of possible (unseen) outcomes of an action.

9. Effect novelty: it may be that an action has been taken, and that its just observed effect or outcome is novel. Again we do not yet capture this.

10. Causal value uncertainty and causal model complexity: it may be that an action has uncertain outcomes, but that this non-determinism can be eliminated, or the uncertainty in the effects reduced. To do this variables and values for them must be identified which tell us which outcomes are more or less likely. In other words if we look in the history of the process we may find additional variables (perhaps latent ones) that improve predictive accuracy. This is statistically like looking for new variables on which to condition action outcome likelihoods.

11. Causal novelty: here is it known that an action which normally has a reliable effect has an unexpected outcome, i.e. there is a surprise, and that there must therefore be a cause of that suprising outcome. This is the case when the operator model of the action in planning is confident about the outcome, but is wrong. In this again we have to look for additional variables which are not currently be in the operator model which explain the surprising outcome.

In the following sections we describe in detail the representations we use in different aspects of the project. We also relate them to this typology. This is the first step towards a unified account of representations for self-understanding.

3 Multi-modal representation of beliefs

Intelligent robots needs to be aware of their own surroundings. This awareness is usually encoded in some implicit or explicit representation of the situated context. Such representation must be grounded in multiple sensory modalities and be capable of evolving dynamically over time. Moreover, it must be able to capture both the rich *relational structure* of the environment and the *uncertainty* arising from the noise and incompleteness of low-level sensory data.

In this section, we present a new framework for constructing rich, multi-modal belief models of the robot's environment. These beliefs unify the representations arising from the different modalities. We start by describing the architecture in which our approach has been integrated, then detail the formal representations used to specify multi-modal beliefs, and finally briefly explain how such beliefs can be constructed from perceptual inputs.

Beliefs also incorporate various contextual information such as spatio-temporal framing, multi-agent epistemic status, and saliency measures. Such rich annotation scheme allows us to easily interface beliefs with high-level cognitive functions such as action planning or communication. Beliefs can therefore be easily referenced, controlled and extended "top-down" by external processes to reach beyond the current perceptual horizon and include past, future or hypothetical knowledge.

The interested reader is invited to look at the extended report entitled "*Belief Modelling for Situation Awareness in Human-Robot Interaction*" attached to this document for more detailed information.

3.1 Architecture

Our approach to rich multi-modal belief modelling is implemented in a specific module called the "*binder*". The binder is directly connected to all subsystems in the architecture (i.e. vision, navigation, manipulation, etc.), and serves as a central hub for the information gathered about the environment.

The core of the binder system is a shared *working memory* where beliefs are formed and refined based on incoming perceptual inputs. Fig. 2 illustrates the connection between the binder and the rest of the architecture.

3.2 Representation of beliefs

Each unit of information manipulated by the binder is expressed as a *probability distribution* over a space of possible values. Such unit of information is called a **belief**.

Beliefs are constrained both *spatio-temporally* and *epistemically*. They include a frame stating where and when the information is assumed to be

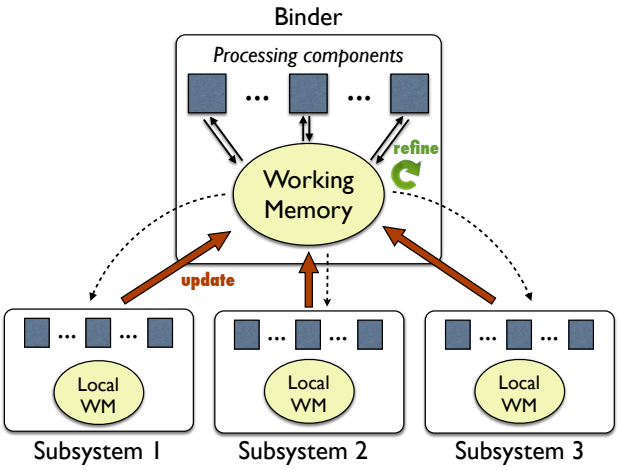


Figure 2: Schema of the cognitive architecture in relation with the binder

valid, and an epistemic status stating for which agent(s) the information holds.

Formally, a **belief** is a tuple $\langle i, e, \sigma, c, \boldsymbol{\delta}, h \rangle$, where i is the belief identifier, e is an epistemic status, σ a spatio-temporal frame, c an ontological category, $\boldsymbol{\delta}$ is the belief content (specified as a probability distribution), and h is the history of the belief.

We describe below each of these components one by one.

3.2.1 Epistemic status e

Interactive robots must be able to distinguish between their own knowledge, knowledge of others, and shared knowledge (common ground). We specify such information in the epistemic status of the belief. For a given agent a, the **epistemic status** e can be either:

- *private*, denoted $\mathsf{K}\{a\}$: private beliefs come from within the agent a. In other words, they are a direct or indirect result of agent a's perception of the environment;
- *attributed*, denoted $\mathsf{K}\{a[b_1, ..., b_n]\}$: Attributed beliefs are beliefs which are ascribed to other agents. They are a's conjecture about the mental states of other agents $b_1, ..., b_n$, usually as a result of a's interpretations of previous communicative acts performed by $b_1, ..., b_n$.
- *shared*, denoted $\mathsf{K}\{a_1, ..., a_m\}$: Shared beliefs contain information which is part of the common ground for the group [11].

3.2.2 Spatio-temporal frame σ

The **spatio-temporal frame** σ defines a contiguous spatio-temporal interval, the nature of which depends on the application domain. In the simplest case, the spatial dimension can be modelled by a discrete set of regions and the temporal dimension via intervals defined on real-valued time points.

It is important to note that beliefs can express past or future knowledge (i.e. memories and anticipations). That is, beliefs need not be directly grounded in the "here-and-now" observations.

3.2.3 Ontological category c

The **ontological category** is used to sort the various belief types which can be created. Various levels of beliefs are defined, from the lowest to the highest abstraction level. Figure 5 illustrates the role of these categories in the belief formation process.

1. The lowest-level type of beliefs is the *percept* (or *perceptual belief*), which is a uni-modal representation of a given entity[1] or relation between entities in the environment. Perceptual beliefs are inserted onto the binder by the various subsystems included in the architecture. The epistemic status of a percept is private per default, and the spatio-temporal frame is the robot's present place and time-point.

2. If several percepts (from distinct modalities) are assumed to originate from the same entity, they can be grouped into a *percept union*. A percept union is just another belief, whose content is the combination of all the features from the included percepts.

3. The features of a percept union can be abstracted using multi-modal fusion and yield a *multi-modal belief*.

4. If the current multi-modal belief (which is constrained to the present spatio-temporal frame) is combined with beliefs encoded in past or future spatio-temporal frames, it forms a *temporal union*.

5. Finally, the temporal unions can be refined *over time* to improve the estimations, leading to a *stable belief*, which is both multi-modal and spans an extended spatio-temporal frame.

3.2.4 Belief content δ

The **distribution** δ defines the possible content values for the belief. In general, each alternative value can be expressed as a (propositional) logical

[1] The term "entity" should be understood here in a very general sense. An entity can be an object, a place, a landmark, a person, etc.

formula. In most practical cases, such formula can be represented as a flat list of features. The feature values can be either discrete (as for categorical knowledge) or continuous (as for real-valued measures). A feature value can also specify a *pointer* to another belief, allowing us to capture the relational structure of the environment we want to model. The resulting relational structure can be of arbitrary complexity.

Discrete probability distributions can be expressed as a set of pairs $\langle \varphi, p \rangle$ with φ a formula, and p a probability value, where the values of p must satisfy the usual constraints for probability values. For continuous distribution, we generally assume a known distribution (for instance, a normal distribution) combined with the required parameters (e.g. its mean and variance).

The distribution can usually be decomposed into a list of smaller distributions over parts of the belief content. This can be done by breaking down the formulae into elementary predications, and assuming conditional independence between these elementary predicates. The probability distribution $\boldsymbol{\delta}$ can then be factored into smaller distributions $\boldsymbol{\delta}_1 ... \boldsymbol{\delta}_n$.

3.2.5 Belief history h

Finally, via the **belief history** h, each belief contains bookkeeping information detailing the history of its formation. This is expressed as two set of pointers: one set of pointers to the belief ancestors (i.e. the beliefs which contributed to the emergence of this particular belief) and one set of pointers to the belief offspring (the ones which themselves emerged out of this particular belief).

3.2.6 Example of belief representation

Consider an environment with a blue mug such as the one pictured in Figure 3. The mug is perceived by the robot sensors (for instance, by one binocular camera). Sensory data is extracted and processed by the sensory subarchitecture(s). A the end of the process, a perceptual belief is created, with four features: object label, colour, location, and height.

Figure 3: A blue mug

Due to the noise and uncertainty of sensory data, the perceived characteristics of the object are uncertain. Let us assume two uncertainties:

- The colour value of the object is uncertain (the vision system hesitates between blue with probability 0.77 and purple with probability 0.22),
- and the recognition of the object itself is also uncertain (the recognised object might be a false positive with no corresponding entity in the real world. The probability of a false positive is 0.1).

Such perceptual belief i would be formally defined as:

$$\langle i, \{\text{robot}\}, \sigma_{[\text{here-and-now}]}, \text{percept}, \boldsymbol{\delta}, h \rangle \tag{1}$$

with a probability distribution $\boldsymbol{\delta}$ containing three alternative formulae φ_1, φ_2 and φ_3. A graphical illustration of the belief i is provided in Figure 4.

We can see in Figure 4 that the formula φ_2 specifies the existence (with probability 0.7) of a blue mug entity of size 11.2 cm, at location k, perceived by the robot in the current spatio-temporal frame ("here-and-now"). Notice that the location is described as a pointer to another belief k. Such pointers are crucial to capture relational structures between entities.

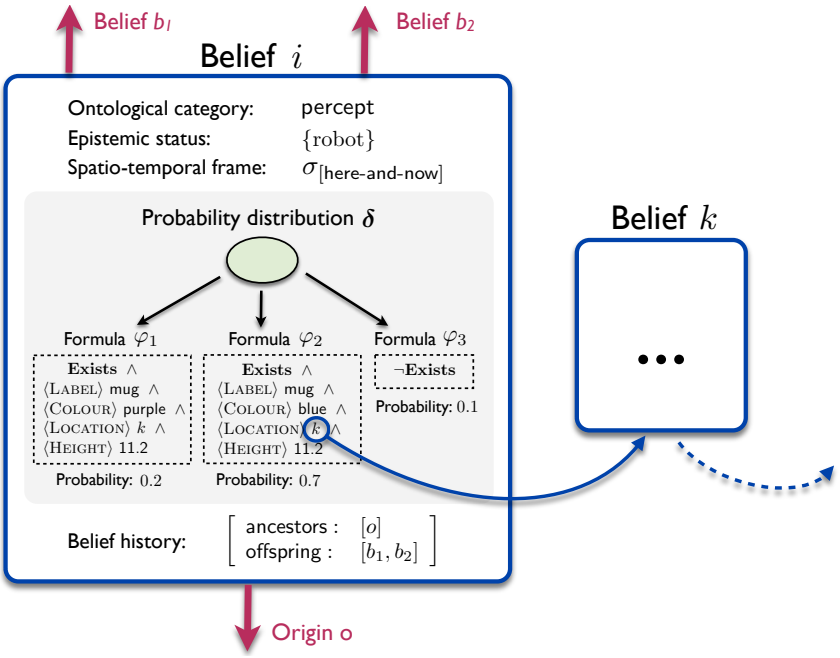


Figure 4: Schematic view of a belief representation.

The belief i also specifies a belief history h. The belief i being a percept, its history is defined as a pointer to a local data structure o in the subarchitecture responsible for the belief's creation. The belief history also contains two pointers b_1 and b_2 to the belief's offspring.

3.3 Bottom-up belief formation

We now turn our attention to the way a belief model can be constructed bottom-up from the initial input provided by the perceptual beliefs. The formation of belief models proceeds in four consecutive steps: (1) *perceptual grouping*, (2) *multi-modal fusion*, (3) *tracking* and (4) *temporal smoothing*. Figure 5 provides a graphical illustration of this process.

The rules governing the construction process are specified using a first-order probabilistic language, *Markov Logic*. Markov Logic is a combination of first-order logic and probabilistic graphical models. Its expressive power allows us to capture both the rich relational structure of the environment and the uncertainty arising from the noise and incompleteness of low-level sensory data. Due to space constraints, we cannot detail the formal properties of Markov Logic here, the interested reader is advised to look at the extended report for further information.

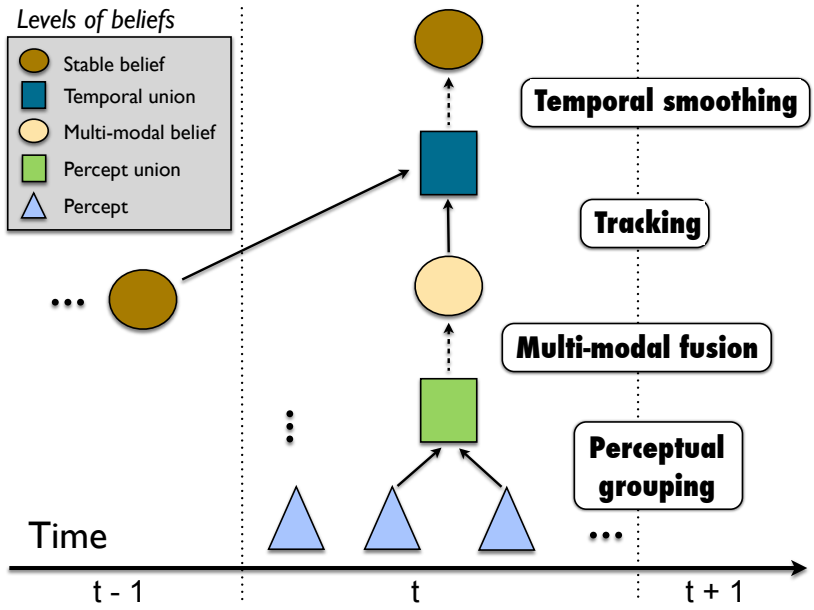


Figure 5: Bottom-up belief model formation.

3.3.1 Perceptual grouping

The first step is to decide which percepts from different modalities belong to the same real-world entity, and should therefore be grouped into a belief. For a pair of two percepts p_1 and p_2, we infer the likelihood of these two percepts being generated from the same underlying entity in the real-world. This is realised by checking whether their respective features *correlate* with each other.

The probability of these correlations are encoded in a Markov Logic Network. From a syntactic point of view, a *Markov logic network* L is simply defined as a set of pairs (F_i, w_i), where F_i is a first-order formula and $w_i \in \mathbb{R}$ is the associated weight of that formula (Markov Logic weights can be directly translated into probability distributions).

The details of such Markov Logic Network are provided in Annex 2. The formulae might for instance express a high compatibility between the haptic feature "shape: cylindrical" and the visual feature "object: mug" (since most mugs are cylindrical), but a very low compatibility between the features "shape: cylindrical" and "object: ball". Eq. (2) illustrates the correlation between the cylindrical shape (`Cyl`) and the object label "mug" (`Mug`).

$$\begin{array}{ll} w_i & \texttt{Shape}(\texttt{P1}, \texttt{Cyl}) \\ w_j & \texttt{Label}(\texttt{P2}, \texttt{Mug}) \\ w_k & \texttt{Shape}(\texttt{x}, \texttt{Cyl}) \wedge \texttt{Label}(\texttt{y}, \texttt{Mug}) \rightarrow \texttt{Unify}(\texttt{x}, \texttt{y}) \end{array} \quad (2)$$

Markov Logic formulae can also express incompatibility between features, for instance between a spherical shape and a object labelled as a mug:

$$w_j \quad \texttt{Shape}(\texttt{x}, \texttt{Spherical}) \wedge \texttt{Label}(\texttt{y}, \texttt{Mug}) \rightarrow \neg\texttt{Unify}(\texttt{x}, \texttt{y}) \quad (3)$$

Additional formulae are used to specify generic requirements on the perceptual grouping process, for instance that `x` and `y` must be distinct beliefs and originate from distinct subarchitectures. The prior probability of a grouping is also specified as a Markov Logic formula.

Metric spatial information is obviously also a crucial component in this perceptual grouping process. Markov Logic being able to handle both discrete and continuous values, one straightforward option would be to encode traditional Bayesian filtering techniques such as Kalman filters into a Markov Logic Network. We still need to investigate, however, what would be the exact consequences of such re-encoding in terms of runtime efficiency. It might be beneficial to perform some preprocessing outside the binder system via fast, dedicated algorithms to speed up the grouping process.

A grouping of two percepts will be given a high probability if (1) one or more feature pairs correlate with each other, and (2) there are no incompatible feature pairs. This perceptual grouping process is triggered at each insertion or update of percepts on the binder (provided the number of modalities in the system > 1). The outcome is a set of possible unions, each of which has an existence probability describing the likelihood of the grouping.

3.3.2 Multi-modal fusion

We want multi-modal beliefs to go beyond the simple superposition of isolated modal contents. Multi-modal information should be *fused*. In other words, the modalities should co-constrain and refine each other, yielding new multi-modal estimations which are globally more accurate than the uni-modal ones.

Multi-modal fusion is also specified in a Markov Logic Network. As an illustration, assume a multi-modal belief $\mathtt{B}$ with a predicate $\mathtt{Position}(\mathtt{B}, \mathtt{loc})$ expressing the positional coordinates of an entity, and assume the value $\mathtt{loc}$ can be estimated via distinct modalities a and b by way of two predicates $\mathtt{Position}_{(\mathtt{a})}(\mathtt{U}, \mathtt{loc})$ and $\mathtt{Position}_{(\mathtt{b})}(\mathtt{U}, \mathtt{loc})$ included in a percept union $\mathtt{U}$.

$$w_i \quad \mathtt{Position}_{(\mathtt{a})}(\mathtt{U}, \mathtt{loc}) \rightarrow \mathtt{Position}(\mathtt{B}, \mathtt{loc}) \tag{4}$$

$$w_j \quad \mathtt{Position}_{(\mathtt{b})}(\mathtt{U}, \mathtt{loc}) \rightarrow \mathtt{Position}(\mathtt{B}, \mathtt{loc}) \tag{5}$$

The weights w_i and w_j specify the relative confidence of the measurements for the modality a and b, respectively.

3.3.3 Tracking

Environments are dynamic and evolve over time – and so should beliefs. Analogous to perceptual grouping which seeks to bind observations over modalities, tracking seeks to bind beliefs *over time*. Both past beliefs (memorisation) and future beliefs (anticipation) are considered. The outcome of the tracking step is a distribution over temporal unions, which are combinations of beliefs from different spatio-temporal frames.

The Markov Logic Network for tracking works as follows. First, the newly created belief is compared to the already existing beliefs for similarity. The similarity of a pair of beliefs is based on the correlation of their content (and spatial frame), plus other parameters such as the time distance between beliefs.

Eq. (6) illustrates a simple example where two beliefs are compared on their shape feature to determine their potential similarity:

$$w_i \quad \mathtt{Shape}(\mathtt{x}, \mathtt{Cyl}) \wedge \mathtt{Shape}(\mathtt{y}, \mathtt{Cyl}) \rightarrow \mathtt{Unify}(\mathtt{x}, \mathtt{y}) \tag{6}$$

If two beliefs $\mathtt{B}_1$ and $\mathtt{B}_2$ turn out to be similar, they can be grouped in a temporal union $\mathtt{U}$ whose temporal interval is defined as $[\mathtt{start}(\mathtt{B}_1), \mathtt{end}(\mathtt{B}_2)]$.

3.3.4 Temporal smoothing

Finally, temporal smoothing is used to refine the estimates of the belief content *over time*. Parameters such as recency have to be taken into account, in order to discard outdated observations.

The Markov Logic Network for temporal smoothing is similar to the one used for multi-modal fusion:

$$w_i \quad \mathtt{Position}_{(\mathtt{t\text{-}1})}(\mathtt{U}, \mathtt{loc}) \rightarrow \mathtt{Position}(\mathtt{B}, \mathtt{loc}) \tag{7}$$

$$w_j \quad \mathtt{Position}_{(\mathtt{t})}(\mathtt{U}, \mathtt{loc}) \rightarrow \mathtt{Position}(\mathtt{B}, \mathtt{loc}) \tag{8}$$

These four consecutive steps are currently being formalised and implemented as part of the binder system.

3.4 Modelling gaps and uncertainty

The framework we outlined here is able to represent various types of uncertainty regarding the state and structure of the environment. Belief content is expressed as a multivariate probability distribution which can capture both **quantitative variable value uncertainty** (via probabilities attached to individual features in a given belief) and **model complexity uncertainty** (via existence probabilities attached to beliefs). **State novelty** can be taken into account by including unknown values into the domain model. Values can be quantitative (e.g. real-valued measures) or qualitative (categorical knowledge). In terms of knowledge types, only **state** values are currently modelled, but we are planning to extend our framework in the future to include not only indexical knowledge, but also events, intentions and plans. The belief modelling framework here is also largely concerned with **contingent** knowledge, i.e. knowledge about the environment as it is now and how it has been recently. It uses general long term semantic knowledge to assist in its inferences, but doesn't represent gaps in that knowledge, this the purpose of cross-modal representations, which we deal with later on. Finally belief modelling captures the epistemic state not just of the agent itself, but also of other agents, of groups of agents, and of groups of agents that share beliefs.

4 Beliefs about situated dialogue

4.1 Introduction

Representations are reflections. They are signs, of *what* an agent understands – and therefore, of *how* an agent understands. Signs *are* processes. This fundamental idea is familiar from semiotics. Its relevance for cognitive systems that are (to be) capable of self-introspection and self-extension is arguably this: Representations can only improve by improving the processes and models that give rise to them. Development in a cognitive system is one part acquiring more representational power, and the ability to interconnect different modalities of meaning. For another part, it is the development of the very ability to compose meaning, and attending to those modal aspects that can drive that composition in context.

In this paper, we describe results and ongoing research in dealing with self-introspection and self-extension in situated dialogue processing. Typically, a gap in dialogue is seen as a lack of understanding, that can lead to a breakdown in communication. Clarification mechanisms can help resolve this, for example through question/answer sub-dialogues. Several discourse theories identify levels at which gaps arise, cf. Allwood [1] or Clark [10]. This can go all the way down to problems in situationally grounding dialogue. We explored the latter problem in more detail in [25]. Ultimately,

though, clarification is just a means to an end. It helps a cognitive system to improve on, or correct, behavior that went wrong. In this paper for WP1, we focus on how gaps can be represented. (In Task 6.3 (DR 6.2, WP6 year 2) we deal with how these representations can then later on drive adaptation at the processing levels.)

We start by looking at logical forms, in §4.2. Logical forms are the basic level at which we represent linguistic meaning. Words and syntactic structure are in some sense all just "artifacts." They are means we use to get us to a first level of meaning for an audio signal. It is at this level that we represent gaps in interpretation. We deal with the structural reflections of typical dialogue phenomena such as ambiguity, incompleteness, even ungrammaticality. But this is just linguistic meaning – meaning in as far as expressed through linguistic means. We do construct meaning in context, using contextually salient information to drive the processes that build up logical forms. At the same time, we need to interpret meaning construed in situated dialogue further, against the background of a collaborative activity.

In §4.3 we then elaborate on how beliefs are bound to the robot's models of the world. We discuss how gaps can be represented as missing information on an open-world assumption, and how we can deal with the dynamics of revising and extending beliefs in situated multi-agent belief models. These models are grounded in the multi-modal belief models we discussed earlier, in §3. This provides us with grounded meaning, which may have gaps in how to understand what is being communicated. This grounding is subject to the uncertainty and incompleteness inherent to a robot's experience of reality. In §3 we describe the probabilistic approach we use to deal with estimation and inference in the context of belief content. Below, we "lift" this to how we can logically reason with (uncertain) beliefs in processing situated dialogue, to determine how to interpret and follow up on what is being talked about.

4.2 Logic forms for representing meaning

We represent linguistic meaning as an ontologically richly sorted, relational structure. This structure is a *logical form* [24, 5] in a decidable fragment of modal logic [7, 2]. The modal-logical aspect of the logic makes it possible to build up structures using named relations. A novel construct, called a "nominal" [7] provides us with an explicit way to index and reference individual structures in a logical form.

The following is an example of a logical form. It expresses a linguistic meaning for the utterance "I want you to put the red mug to the right of the ball." Each node in the logical form has a nominal, acting as unique identifier for that node. We associate the nominal with an ontological sort, e.g. $p1 : action - motion$ means that $p1$ is of sort $action - motion$, and a proposition, e.g. **put** for $p1$. We connect nodes through named relations.

These indicate how the content of a single node contributes to the meaning of the whole expression. For example, "you" ($y1$) both indicates the one whom something is wanted of (*Patient*-relation from $w1$), and the one who is to perform the put action (*Actor*-relation from $p1$). Nodes carry additional features, e.g. $i1$ identifies a singular person.

$$
\begin{aligned}
&@_{w_1}\,\text{:cognition}(\textbf{want} \wedge \langle \textsc{Mood} \rangle\, ind \wedge \langle \textsc{Tense} \rangle\, pres \wedge \\
&\quad \langle \textsc{Actor} \rangle\, (i_1 : \text{person} \wedge \textbf{I} \wedge \langle \textsc{Num} \rangle\, sg) \wedge \\
&\quad \langle \textsc{Event} \rangle\, (p_1 : \text{action-motion} \wedge \textbf{put} \wedge \\
&\qquad \langle \textsc{Actor} \rangle\, y_1 : \text{person} \wedge \\
&\qquad \langle \textsc{Patient} \rangle\, (m_1 : \text{thing} \wedge \textbf{mug} \wedge \\
&\qquad\quad \langle \textsc{Delimitation} \rangle\, unique \wedge \langle \textsc{Num} \rangle\, sg \wedge \langle \textsc{Quantification} \rangle\, specific \wedge \\
&\qquad\quad \langle \textsc{Modifier} \rangle\, (r_1 : \text{q-color} \wedge \textbf{red})) \wedge \\
&\qquad \langle \textsc{Result} \rangle\, (t_1 : \text{m-whereto} \wedge \textbf{to} \wedge \\
&\qquad\quad \langle \textsc{Anchor} \rangle\, (r_2 : \text{e-region} \wedge \textbf{right} \wedge \\
&\qquad\quad \langle \textsc{Delimitation} \rangle\, unique \wedge \\
&\qquad\quad \langle \textsc{Num} \rangle\, sg \wedge \\
&\qquad\quad \langle \textsc{Quantification} \rangle\, specific \wedge \\
&\qquad\quad \langle \textsc{Owner} \rangle\, (b_1 : \text{thing} \wedge \textbf{ball} \wedge \\
&\qquad\quad \langle \textsc{Delimitation} \rangle\, unique \wedge \langle \textsc{Num} \rangle\, sg \wedge \langle \textsc{Quantification} \rangle\, specific)))) \wedge \\
&\quad \langle \textsc{Patient} \rangle\, (y_1 : \text{person} \wedge \textbf{you} \wedge \langle \textsc{Num} \rangle\, sg) \wedge \\
&\quad \langle \textsc{Subject} \rangle\, i_1 : \text{person})
\end{aligned}
$$

Propositions and relations in such a representation are instances of concepts. This makes it possible for us to interpret logical forms further using ontological reasoning. We use this possibility in reference resolution, and in relating meaning representations to interpretations formed outside the dialogue system. Furthermore, the combination of sorting and propositional information provides a basic way of representing gaps. Both can be underspecified: An indicated sort may vary in specificity, and a lack of a proposition indicates (under an open-world assumption) a gap in information.

The relational nature of our representations provides us with several consequences. We build up our representations from elementary propositions as we illustrated above – sorted identifiers and propositions, features, and relations. An interpretation is thus simply a conjunction of such elementary propositions, and the more we can connect those elementary propositions, the more complete our interpretation becomes. This has several important consequences. For one, it means that we can break up linguistic meaning into small, interconnected parts. Each elementary proposition acts as a sign, signifying a particular meaningful dimension of the whole it is connected to (by virtue of interconnected identifiers). Second, elementary propositions make it relatively straightforward to represent partial interpretations. For example, for "take the red ..." receives the following interpretation:

$@t_1$:action-motion(**take** $\wedge$ $\langle$Mood$\rangle$ *imp* $\wedge$ $\langle$Tense$\rangle$ *pres* $\wedge$
$\langle$Actor$\rangle$(a_1 : entity $\wedge$ **addressee**) $\wedge$
$\langle$Patient$\rangle$ (m_1 : thing $\wedge$
$\langle$Delimitation$\rangle$ *unique* $\wedge$ $\langle$Num$\rangle$ *sg* $\wedge$ $\langle$Quantification$\rangle$ *specific* $\wedge$
$\langle$Modifier$\rangle$ (r_1 : q-color $\wedge$ **red**))
$\langle$Subject$\rangle$ a_1 : entity)

The interpretation shows more than just the content for the three words. It also shows that "red" is expected to be the color of the "thing" which is supposed to be taken.

Third, characteristic for language is that it presents many ways in which we can say things – and interpret them. This inevitably means that we will usually get not just one, but multiple alternative interpretations for an utterance. To keep ambiguity to a minimum, we should look at to what extent these interpretations are indeed different. Where they show overlaps, we should ideally have to deal with those identical parts only once. Using relational structure and elementary propositions enables us to do so. We represent alternative interpretations as alternative ways in which we can connect content, whereas identical content across interpretations is represented once. The procedure to create such "condensed" representations is called *packing*, after [31, 9]. Figure 6 illustrates the development of the packed packed representation for "here is the ball". At the first step ("take"), 9 logical forms are packed together, with two alternative roots, and several possible ontological sorts for the word "here". The second step reduces the number of alternative interpretations to one single logical form, rooted on the verb "be" with a "presentational" ontological sort. The possible meanings for the determiner is expressed at the dependent node of the "Presented" relation. At this point we have an *overspecified* meaning. Although the delimination is unique, we cannot tell at this point whether we are dealing with a singular object, or a non-singular (i.e. plural) object – all we know it has to be one or the other. This becomes determined in the fourth step ("here is the ball").

Detailed accounts of how contextual information can be used to guide the construction of logical forms in situated dialogue are provided in [26, 27].

4.3 Grounding meaning in belief models

In the previous section we discussed how linguistic meaning can be seen as a relational structure over small signs. We gradually build up such a structure, using incremental processing that is guided by what is currently contextually salient. A structure can indicate what information may be still lacking (sortal or propositional), what further information is expected, and what alternative ways there appear to be to connect signs. In the current section, we present an approach to how situated beliefs are formed. We exploit the

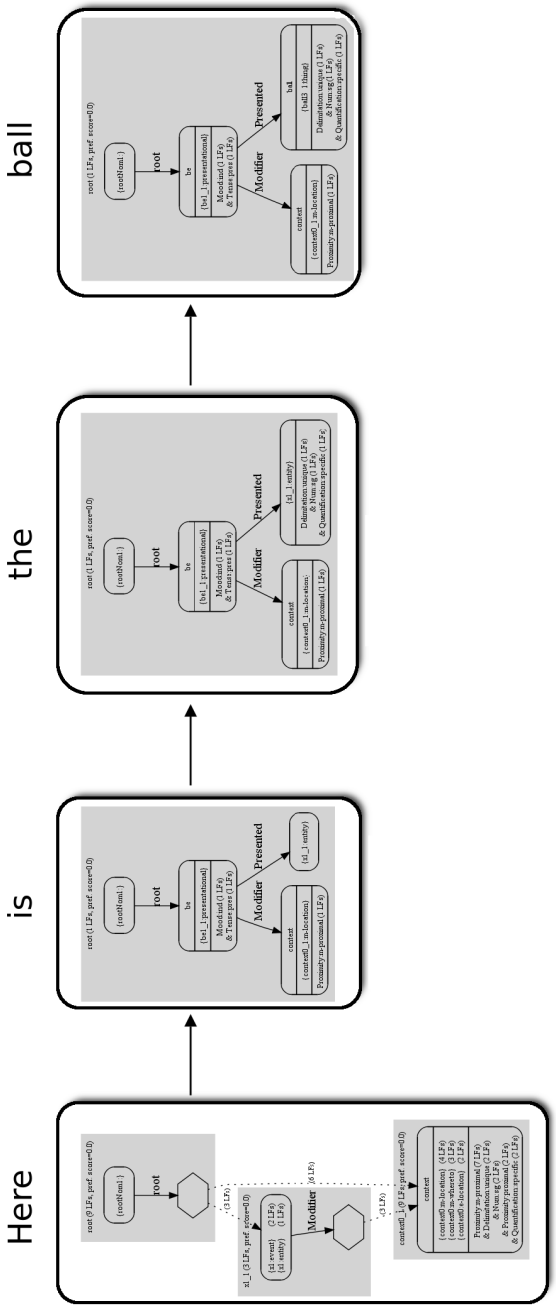


Figure 6: Example of incremental parsing and packing of logical forms, for the utterance "Here is the ball". Four steps are shown.

idea of small signs in grounding linguistic meaning in the cognitive system's models of the world, and forming relational belief structures.

A belief is a formula $\mathsf{K}e/\sigma : \phi$ that consists of three parts: a *content formula* ϕ from a *domain logic* $\mathcal{L}_{\mathsf{dom}}$, the assignment e of the content formula to agents, which we call an *epistemic status* and the *spatio-temporal frame* σ in which this assignment is valid.

We distinguish three classes of epistemic statuses, that give rise to three classes of beliefs:

- **private** belief of agent a, denoted $\{a\}$, comes from *within* the agent a, i.e. it is an interpretation of sensor output or a result of deliberation.
- a belief **attributed** by agent a to other agents $b_1, ..., b_n$, denoted $\{a[b_1, ..., b_n]\}$, is a result of a's deliberation about the mental states of $b_1, ..., b_n$ (e.g. an interpretation of an action that they performed).
- a belief **shared** by the group of agents $a_1, ..., a_m$, denoted $\{a_1, ..., a_m\}$, is common ground among them.

A spatio-temporal frame is a contiguous spatio-temporal interval. The belief is only valid in the spatio-temporal frame σ and frames that are subsumed by σ. This way, spatio-temporal framing accounts for situatedness and the dynamics of the world. The underlying spatio-temporal structure may feature more complex spatial or temporal features.

Finally, the domain logic $\mathcal{L}_{\mathsf{dom}}$ is a propositional modal logic. We do not require $\mathcal{L}_{\mathsf{dom}}$ to have any specific form, except for it to be sound, complete and decidable.

Multiple beliefs form a *belief model*. A belief model is a tuple $\mathbf{B} = (A, S, K, F)$ where A is a set of agents, S is a set of spatio-temporal frames, K is a set of beliefs formed using A and S and $F \subseteq K$ is a set of *activated* beliefs.

Belief models are assigned semantics based on a modal-logical translation of beliefs into a poly-modal logic that is formed as a fusion of $\mathrm{KD45}^{\mathsf{C}}_{A}$ (doxastic logic with a common belief operator [15]) for epistemic statuses, K4 for subsumption-based spatio-temporal reasoning and $\mathcal{L}_{\mathsf{dom}}$ for content formulas. This gives us a straightforward notion of belief model consistency: a belief model is consistent if and only if its modal-logical translation has a model.

The belief model keeps track of the beliefs' evolution in a directed graph called the *history*. The nodes of the history are beliefs and operations on the belief model (such as *retraction*) with (labeled) edges denoting the operations's arguments. The nodes that are beliefs and have no outcoming edges form a consistent, most recent belief model.

The possibility for a belief to evolve is fundamental to attaining and maintaining common ground. With this evolution comes an increase in

logical implicational power. A shared belief of a group G that ϕ implies all private beliefs and all possible attributed beliefs that ϕ within that group. For example, if ϕ is common ground between the human user, h, and robot, r, then (i) implies (ii):

$$\underbrace{\mathbf{B} \models \mathsf{K}\{\mathsf{r},\mathsf{h}\}/\sigma : \phi}_{\text{(i)}} \quad \Rightarrow \quad \underbrace{\begin{array}{ll} \mathbf{B} \models \mathsf{K}\{\mathsf{r}\}/\sigma : \phi & \\ \mathbf{B} \models \mathsf{K}\{\mathsf{r}[\mathsf{h}]\}/\sigma : \phi & \\ \mathbf{B} \models \mathsf{K}\{\mathsf{h}\}/\sigma : \phi & * \\ \mathbf{B} \models \mathsf{K}\{\mathsf{h}[\mathsf{r}]\}/\sigma : \phi & * \end{array}}_{\text{(ii)}}$$

Since (i) and (ii) are inferentially equivalent within belief models, the relation is in fact equivalence. If (ii) holds in the belief model $\mathbf{B}$, it also satisfies (i).

However, the agents' private and attributed beliefs cannot be observed by other agents, they are not ominiscient. The beliefs above marked by asterisk (*) cannot be present in the robot's belief model. The validity of such beliefs can only be *assumed*. An invalidation of the assumptions then invalidates the premise (ii) and thus the conclusion (i). As long as they are not invalidated, agents may act upon them: they may *assume* that common ground has been attained.

But how can these assumptions be in principle mandated or falsified? Given a communication channel C, we consider a class of protocols P_C that supply the means for falsification of the assumptions. If these means are provided, then the protocol is able to reach common ground. We assume that the agents are faithful to Grice's Maxim of Quality [19], i.e. that they are truthful and only say what they believe to be true and for what they have evidence.

4.4 Conclusions

Looking back at the discussions in §3 and §4, from the viewpoint of cognitive systems, we can say simply this: Gaps can arise everywhere. Literally. No aspect is too small for a system not to have a hole in it, to lack a certain degree of understanding. Representations can indicate this lack, and processing can help overcome this lack by combining with other sources of information (the multi-sensor fusion hypothesis), by filtering it out (the attention hypothesis) – or by self-extension to ultimately achieve better interpretations (the continual development hypothesis). Not just at representing things better, but processing it better.

We presented here parts of the representational side of self-understanding and self-extension, focusing on situated dialogue processing. Connecting our representations to the typology presented in §2 (cf. also [25]), the gaps and uncertainty we discussed here primarily concern **state variable uncertainty**, and **state model complexity**. In the linguistic representations

per se we handle uncertainty against a model, either in variable assignment (e.g. ASR hypotheses) or in how to connect hypotheses (e.g. partial parses). We do not yet deal with unknown words or syn-semantic constructions (but see tasks later in CogX). At the same time, based on further interpretation of a representation of linguistic meaning against a situated, multi-agent belief model, we can see further gaps and uncertainties arise. Beliefs can express uncertainty about how to ground references to observations of the environment (**structural variable values**), or uncertainty about the truth of statements attributed to other agents (i.e. **multi-agent epistemic status** in Figure 2). Finally, dialogue has been used in e.g. George, and in precursors to the Dora system, to express uncertainty about state observations, and to sollicit information from a user to help resolve these uncertainties.

In WP6, we look at how we can take the self-understanding aspect in linguistic representation then to the self-extending aspect in processing, to yield adaptive dialogue processing.

5 Beliefs about Space

Spatial knowledge constitutes a fundamental component of the knowledge base of a mobile agent, such as Dora, and many functionalities directly depend on the structure of the spatial knowledge representation, ranging from navigation, over spatial understanding and communication. This section explains how different domains of spatial knowledge are represented in our system. The representation structures and abstracts what is known, and it also represents uncertainty and knowledge gaps explicitly.

The representation is designed for representing complex, cross-modal, spatial knowledge that is inherently uncertain and dynamic. Therefore, it is futile to represent the world as accurately as possible. A very accurate representation must be complex, require a substantial effort to synchronize with the dynamic world and still cannot guarantee that sound inferences will lead to correct conclusions [13]. Our primary assumption is that the representation should instead be minimal and inherently coarse and the spatial knowledge should be represented only as accurately as it is required to provide all the necessary functionality of the system. Furthermore, redundancy is avoided and whenever possible and affordable, new knowledge should be inferred from the existing information.

In our system, spatial knowledge is represented in multiple layers, at different levels of abstraction, from low-level sensory input to high level conceptual symbols. Information is abstracted as much as possible in order to make it robust to the dynamic changes in the world and representations that are more abstract are used for longer-term storage. At the same time, knowledge extracted from immediate observations can be much more accurate (e.g. for the purpose of visual servoing). In other words, the agent uses

the world as an accurate representation whenever possible. It is important to mention that rich and detailed representations do not constitute a permanent base for more abstract ones (as is the case in [39]). Similarly to abstraction levels, space is represented on different spatial scales from single scenes to whole environments. Moreover, space is discretized into a finite number of spatial units. Discretization drastically reduces the number of states that have to be considered e.g. during the planning process [20] and serves as a basis for higher level conceptualization [39].

5.1 Structure of the Representation

Figure 7 on the following page gives a general overview of the structure of the representation. It is sub-divided into layers of specific representations. We distinguish between four layers which focus on different aspects of the world, abstraction levels of the spatial knowledge and different spatial scales. Moreover, each layer defines its own spatial entities and the way the agent's position in the world is represented. At the lowest abstraction level we have the sensory layer which maintains an accurate representation of the robot's immediate environment extracted directly from the robot's sensory input. Higher, we have the place and categorical layers. The place layer provides fundamental discretisation of the continuous space explored by the robot into a set of distinct *places*. The categorical layer focuses on low-level, long-term categorical models of the robot's sensory information. Finally, at the top, we have the conceptual layer, which associates human concepts (e.g., object or room category) with the categorical models in the categorical layer and groups places into human-compatible spatial segments such as rooms.

The following subsections provide additional details about each of the layers and their instantiations within our system. For a detailed theoretical discussion on those principles and optimal implementations, we refer the reader to [34].

5.1.1 Sensory Layer

In the sensory layer, a detailed model of the robot's immediate environment is represented based on direct sensory input as well as data fusion over space around the robot. The sensory layer stores low-level features and landmarks extracted from the sensory input together with their exact position with respect to the robot. The uncertainty associated with the pose of the robot and the location of all landmarks in the local surrounding is explicitly represented using a multivariate Gaussian distribution [35, 17]. Landmarks that move beyond a certain distance are forgotten and replaced by new information. Thus, the representation in the sensory layer is akin to a sliding window, with robot-centric and up-to-date direct perceptual information. It is also essentially bottom-up only, though directives and criteria, such as

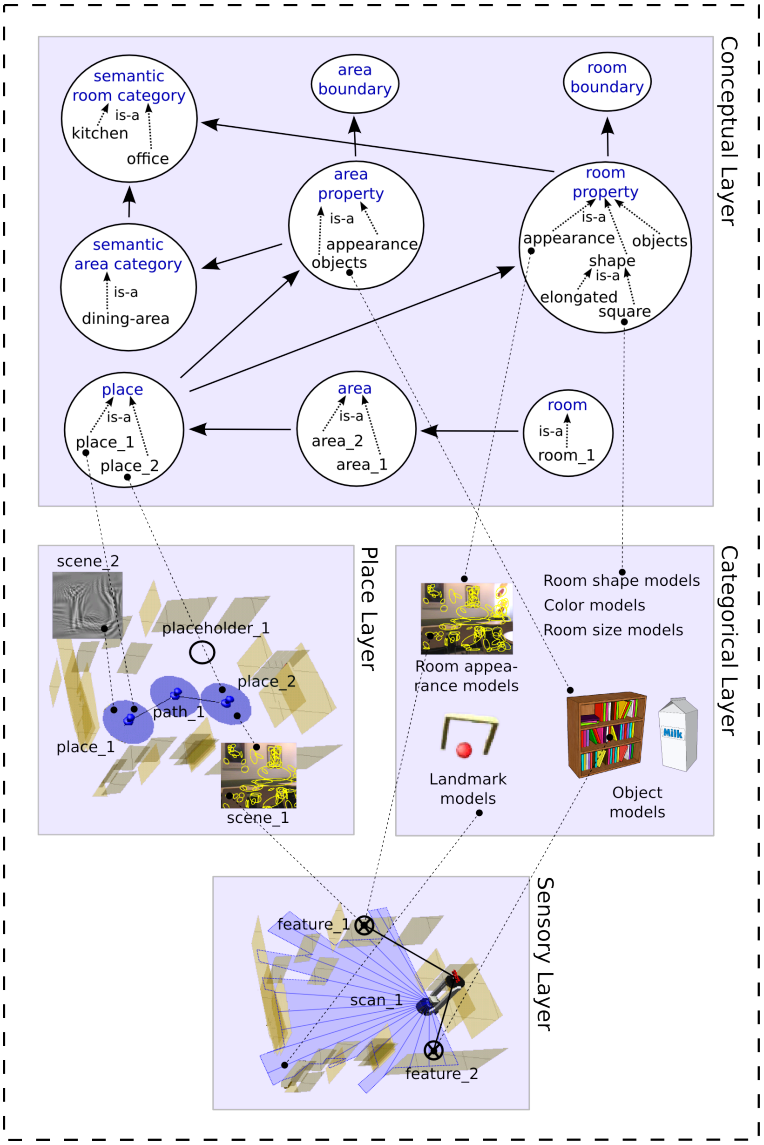


Figure 7: The layered structure of the spatial representation. The position of each layer within the representation corresponds to the level of abstraction of the spatial knowledge.

guiding the attentional process, may be imposed from upper layers. It can contain data of both a 2D and 3D nature.

In addition to the landmark based representation, *local gridmaps* are maintained, centered on each Place. Each cell in the grid map can be in one of three possible states; `occupied, free, unknown`. Each cell is initialized as being `unknown`. The grid maps thus represents the structural knowledge with an explicit representation for the gap in spatial knowledge corresponding to the what part of space is unexplored. Figure 8 on the next page shows two examples of local grid maps of adjacent Places. The white area is `free` space: open areas swept out by the robot's laser scanner from within the Place. The black regions represent obstacles, i.e. `occupied,` space and gray denotes unexplored space (`unknown`), which constitutes a knowledge gap.

The visual object search routine maintains hypotheses about existence of objects of specific categories at specific locations using a probabilistic grid representation [3]. Each cell in the grid contains two peices of information, i) the estimated probability of the center of a certain object being in that cell given all the accumulated evidence, $p(C_{i,j}|Z)$, and ii) whether the object recognition algorithm has been run looking at that cell, $S_{i,j}$. Here $C_{i,j}$ denotes the event that object i has its center in cell j and Z denotes the evidence received so far. Evidence comes both in the form of the outcome of visual recognition routines performed from certain given positions and structural knowledge such as certain cells being occupied or not and whether it contains planar surfaces which afford placing objects on top. Besides the high level gaps regarding the position or existence of the object the robot is looking for, there are two types of gaps in knowledge that are explicitly represented in the context of object search. The first one is unexplored space, described above, representing a gap in knowledge about the structural information of space which provides very strong clues in the search process. The second gap relates to the part of space that the robot has not yet visually searched.

5.1.2 Place Layer

The place layer is responsible for the fundamental, bottom-up discretisation of continuous space. In the place layer, the world is represented as a collection of basic spatial entities, the *Places*, as well as their spatial relations. Each place is defined in terms of features that are represented in the sensory layer, but also spatial relations to other places. The relations do not have to be globally consistent as long as they are preserved locally with sufficient accuracy. The representation of places in the place layer persists over long term; however, knowledge that is not accessed or updated can be compressed, generalized and finally forgotten. In more detail, consider a set $\{f_i\}_{i=1}^{n_f}$ of features f_i defined as

$$f_i\left(\boldsymbol{x}, t\right) : \mathcal{C} \times \mathbb{R} \rightarrow \mathcal{F}_i \in \mathbb{R}^n \tag{9}$$

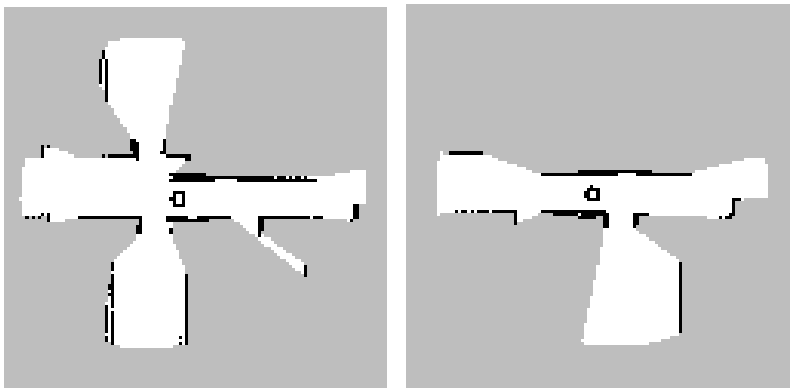


Figure 8: Example of exploration local gridmaps with white: `free`, black: `occupied` and grey: `unknown` space. The position of teh robot is given by the cirtlce in the center.

where $\mathcal{C}$ represents the configuration space of the agent, $t \in \mathbb{R}$ represents time, and $\mathcal{F}_i$ is the range of values of the feature f_i. The feature space is defined as

$$\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \times \ldots \times \mathcal{F}_{n_f}. \tag{10}$$

Places are defined, in part, as non-overlapping collections of tuples of features $(\zeta_1, \ldots, \zeta_{n_f})$. Intuitively, this leads to a division of metric space into regions based on properties such as appearance. As such, two distant disconnected regions could share similar properties. Additional power to distinguish between such regions can be attained using relations between regions such as the adjacency relation for which $\mathcal{R}_i = \{1, 0\}$. By combining the features with the spatial relations we can introduce the *place descriptor space*

$$\mathcal{D} = \mathcal{F} \times \mathcal{R}_1 \times \mathcal{R}_2 \times \ldots \times \mathcal{R}_{n_r}, \tag{11}$$

in which each tuple $D = (\zeta_1, \ldots, \zeta_{n_f}, \rho_1, \ldots, \rho_{n_r})$ of the feature values and relation values $\rho_i = r_i(\boldsymbol{x}, t)$ corresponds to a single point. The *place map* is defined as a set

$$\mathcal{M} = \left\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{n_p}\right\} \tag{12}$$

of *places* $\mathcal{P}_i$ defined such that $\forall_i \mathcal{P}_i \neq \emptyset$ and $\forall_{i \neq j} \mathcal{P}_i \cap \mathcal{P}_j = \emptyset$. If features and relations are time-invariant, the extents of places will be time-invariant as well. Typically, the nature of relations will mean that they are time-invariant as long as the features are. The extents of place is however subject to uncertainty due to measurement noise. The closer to the border between places the larger the uncertainty will typically be as to which place the robot is currently in.

The place layer also defines *paths* between the places. The semantic significance of a path between two places is the possibility of moving directly between one and the other. This does not necessarily imply that the robot

has traveled this path previously. A link might be created for unexplored place e.g. based on top-down cues resulting from the dialogue with the user (e.g. when the robot is guided and the user indicates part of the environment that should be of interest to the robot, but not immediately).

In addition, the place layer explicitly represents *gaps in knowledge about explored space*. Space that has not yet been explored by the robot has no places in it. Therefore, tentative places are generated, which the robot would probably uncover if it moved in a certain direction. These hypothetical places allow for reasoning about unknown space, and for planning and executing exploratory activities. They are annotated as *placeholders* to keep them apart from ordinary, actual places, but are otherwise identically represented and interconnected. For an illustrative example of several places and placeholders identified during spatial exploration, see Figure 7 on page 29. Placeholders are generated wherever there is a frontier between explored and unexplored space near the current Place. The robot can then explore that frontier by moving towards the placeholder, possibly giving rise to a new Place there. Two quantitative measures are associated with each placeholder providing an estimate of information gain related to each exploration task. They are computed from the local grid map associated with the current Place, and are used by the motivation system. The measures used are the *coverage estimate* (CE) and the *frontier length estimate* (FLE), cf. Figure 9 on the next page. The former is obtained by measuring the free space visible from the current node and not near to any existing node, and assigning it to the closest placeholder. This heuristically estimates the number of new places that would result from exploring that direction. The FLE is analogously extracted from the length of the border to unknown space. By prioritising these two measures differently, the motivation mechanism can produce different exploratory behaviours.

5.1.3 Categorical Layer

The categorical layer contains long-term, low-level representations of categorical models of the robot's sensory information. The knowledge represented in this layer is not specific to any particular location in the environment. Instead, it represents a general long-term knowledge about the world at the sensory level. For instance, this is the layer where multi-modal models of landmarks, objects or appearance-based room category or other properties of spatial segments such as shape, size or color are defined in terms of low-level features. The position of this layer in the spatial representation reflects the assumption that the ability to categorise and group sensory observations is the most fundamental one and can be performed in a feed-forward manner without any need for higher-level feedback from cognitive processes.

The categorical models stored in this layer give rise to concepts utilised

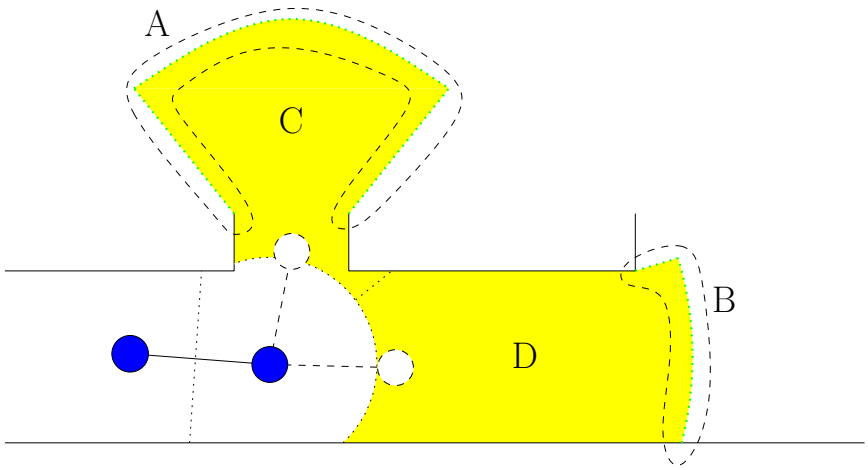


Figure 9: Placeholder creation. Dashed circles are placeholders, each representing one placeholder. A and B are frontier length estimates, C and D are coverage estimates for the respective placeholders.

by higher-level layers. In many cases complex models are required that can only be inferred from training data samples. In case of models that correspond to human concepts, they can be learnt in a supervised fashion, using a top-down supervision signal. Due to the high complexity of the models, unused knowledge might be compressed and generalized.

More concretely, the categorical models are built in a disciriminative, supervised fashion using Support Vector Machines [12]. For simplicity, let as assume that we deal with two categories only, e.g. representing a room and a corridor. In such case, we forumlate the problem of separating a set of training data $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$ into two classes, where $\boldsymbol{x}_i \in \Re^N$ is a feature vector and $y_i \in \{-1, +1\}$ its class label. If we assume that the two classes can be separated by a hyperplane in some Hilbert space $\mathcal{H}$, then the optimal separating hyperplane is the one which has maximum distance to the closest points in the training set resulting in a discriminant function

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{13}$$

The classification result is then given by the sign of $f(\boldsymbol{x})$.

The extension of SVM to multi class problems can be done in several ways. Here we employ the One-against-All (OaA) strategy. If M is the number of classes, M SVMs are trained, each separating a single class from all remaining classes. The decision is then based on the algebraic distance of the classified sample to each hyperplane $f_h(\boldsymbol{x})$ and the final output is the class corresponding to the hyperplane for which the distance is largest.

Discriminative classifiers do not provide any out-of-the-box solution for estimating confidence of the decision; however, it is possible to derive confidence information and hypotheses ranking from the distances between the

samples and the hyperplanes [32]. It is straightforward to extend the OaA multi-class strategy so that additional information about the decision becomes available. Let us define the score $V_h(\boldsymbol{x})$ to be equal to the distance from the average distance of the training samples to the hyperplane:

$$V_h(\boldsymbol{x}) = \left| \widehat{f}_h - f_h(\boldsymbol{x}) \right| .$$

Thus, we do not measure how far the test sample is from the hyperplane, but how close it is to the training data belonging to one of the classes. The best hypothesis can be determined by the following rule:

$$h^* = \underset{j=1...M}{\operatorname{argmin}} \{V_h(\boldsymbol{x})\} . \tag{14}$$

If now we think of the confidence as a measure of unambiguity of the decision, we can define it as follows

$$C(\boldsymbol{x}) = \min_{h=1...M,\ h \neq h^*} \{V_h(\boldsymbol{x})\} - V_{h^*}(\boldsymbol{x}). \tag{15}$$

The value $C(\boldsymbol{x})$ can be thresholded for obtaining a binary confidence information. Moreover, we can order the hypotheses using the values of $V_h(\boldsymbol{x})$.

In our framework, the categorical models might be derived from multiple modalities (e.g. geometrical information extracted from laser range scans and appearance-based information extracted from vision). In order to integrate the different modalities, we used the SVM-DAS integration scheme [33]. The basic idea is to accumulate the outputs of a multi-class discriminative classier obtained for various cues with a complex, possibly non-linear function. Specifically, the accumulation function is given as:

$$V_k^{\Sigma P}(\boldsymbol{I}) = \sum_{i=1}^{m} \alpha_i^k y_i K(\boldsymbol{V}_i, \boldsymbol{V}) + b^k,\ k = 1, \ldots, K,$$

where $\boldsymbol{V}$ is a vector containing all the outputs for all cues:

$$\boldsymbol{V} = \left[\{V_h^1(T_1(\boldsymbol{I}))\}_{h \in H_1}, \ldots, \{V_h^P(T_P(\boldsymbol{I}))\}_{h \in H_P} \right] .$$

The final decision as well as confidence estimates can be obtained using the same method as for a single-cue model.

Examples of gaps in categorical knowledge arises, for example, when the user talks about an object yet uknown to the robot. This means that there is no model for the object. Actively acquiring such a model is an example of self-extension. Depending on what innate knowledge the robot is provide with, there may also be gaps/uncertainty in the number of room categories that are present and models for these.

There is a close connection between the integration of information from multiple sensory modalities that goes on in the categorical layer and the

more general multi-modal belief processing described in section 3 on page 12. In principle the estimation of room categories could be done at that level but it is kept within the spatial representation because, i) it is that the nothing but the end result is used by any component outside the spatial representation and ii) there is plenty of domain knowledge that would have to be propagated to the more general processing levels to be able to perform the fusion.

5.1.4 Conceptual Layer

The conceptual layer provides an ontology that represents taxonomy of the spatial concepts and properties of spatial entities that are linked to the low-level categorical models stored in the categorical layer. This associates semantic interpretations with the low-level models and can be used to specify which properties are meaningful e.g. from the point of view of human-robot interaction. Moreover, the conceptual layer represents relations between the concepts and instances of those concepts linked to the spatial entities represented in the place layer. This makes the layer central for verbalization of spatial knowledge and interpreting and disambiguating verbal expressions referring to spatial entities.

The second important role of the conceptual layer is to provide definitions of the spatial concepts related to the semantic segmentation of space based on the properties of segments observed the environment. A building, floor, room or area are examples of such concepts. The conceptual layer contains information that floors are usually separated by staircases or elevators and that rooms usually share the same general appearance and are separated by doorways. Those definitions can be either given or learned based on asserted knowledge about the structure of a training environment introduced to the system.

Finally, the conceptual layer provides definitions of semantic categories of segments of space (e.g. areas or rooms) in terms of the values of properties of those segments. These properties can reflect the general appearance of a segment as observed from a place, its geometrical features or objects that are likely to be found in that place.

The representation underlying the conceptual map is an OWL-DL ontology[2], consisting of a taxonomy of concepts ($TBox$) and the knowledge about individuals in the domain ($ABox$), cf. Figure 7 on page 29, cf. [39]. Here is an example of a *concept definition* in the current implementation which defines a kitchen as a room that contains at least two typical objects:

$$\mathsf{Kitchen} \equiv \mathsf{Room} \sqcap \geq \mathsf{2contains.KitchenObject}$$

Besides the usual inferences performed by the OWL-DL reasoner, namely *subsumption checking* for concepts in the TBox (i.e., establishing subclass/-

[2] `http://www.w3.org/TR/owl-guide/`

superclass relations between concepts) and *instance checking* for ABox members (i.e., inferring which concepts an individual instantiates), an additional *rule engine* is used to maintain a symbolic model of space under *incomplete* and *changing* information.

The discrete places from the place layer and their adjacency are the main pieces of knowledge that constitute the input for that reasoning. One, it maintains a representation that groups places into rooms. Furthermore, using observations (visually detected objects, appearance- and geometry-based room categories) it can infer human-compatible concepts for a room, and raise expectations about which other kinds of objects are proto-typically likely to be present. The ongoing construction of the conceptual map is potentially nonmonotonic. The overall room organisation may be revised on the basis of new observations. The further association between room concepts and salient, proto-typical object types is established through the "locations" table of the OpenMind Indoor Common Sense[3] database by Honda Research Institute USA Inc.

In the current implementation, the conceptual layer can be used to determine *knowledge gaps in the categorisation of rooms.* It is considered a gap in knowledge if for a given room (i.e., an instance of PhysicalRoom) its basic level category is unknown. This is assumed to be the case if no more specific concept than PhysicalRoom (i.e., Office or Kitchen, cf. Figure 7 on page 29) can be inferred for the individual. This knowledge gap persists until the robot has gathered enough evidence (i.e., contained objects) for inferring a subconcept.

5.1.5 Relation to typology

This section will illustrate the type of uncertainties and gaps that are present in the spatial representation in a simple example. In the example the robot is given a tour of the environment and then it is asked to find a certain object.

Assuming that the robot starts with no prior specific information about the environment the sensory layer would contain a grid map covering the region around the robot with all of space marked as unknown. There would be no landmarks detected yet so the detailed local spatial representation would only contain the robot's pose. The robot then starts to gather information from its sensors (laser scanner and cameras). Cells in the grid map representing what parts of space has been explored will gradually change state from unexplored to free/occupied. Landmarks will be detected and their pose estimated. The uncertain landmark and robot poses together with the state of the cells in the exploration graphs are examples of **state variable uncertainty** as defined in the typology in Section 2 on page 8. The robot

[3] `http://openmind.hri-us.com/`

does not know in advance how large the environment is and how many rooms and objects there are which is an example of **uncertainty about state model complexity**. After some time the scene will change and the robot will add a new Place to the place layer. More and more places will be added and paths between them as well. The initially unknown connectivity of places is an example of uncertainty in **structural model complexity**. Navigating between places also include a level of **effect value uncertainty**.

Based on low-level information as well as for example the presence of objects various properties will be associated with the Places. These properties will be uncertain (**state variable uncertainty**). In the current system the number of categories is known in advance but extensions would allow to incorporate **uncertainty about state model complexity** here as well. Based on the properties of the places, the places are grouped into rooms on the conceptual level in the spatial representation. A soon as a new room instance is created a gap regarding the category of that room appear, ie whether it is a kitchen, or an office, etc.

Assume now that the user asks the robot to find a certain object. At this point we assume that a model of the object is given so there is no uncertainty there. Also assume that the robot does not know where the object is already. This will lead to a gap in knowledge regarding the position of the object. Based on information regarding the statistical relation between objects and other structures a prior for the location of the object can be calculated. This is an example of uncertainty in the **structural variable value**. The search process involves both uncertainty in the **causal model complexity** as there are currently many unmodelled variables that effect the result such as occlusions, ambient lighting conditions but also **causal value uncertainty** since the uncertainty in the effects of running the object recognizer on images from different view points can be reduced by gathering more information about the location of the object.

6 Beliefs about vision

The purpose of vision in the context of an embodied, situated agent is to purposefully interpret the visual scene. There will typically be a task or tasks to pursue and embodiment will define certain constraints. Together these form a dynamically changing context in which vision operates.

For the remainder of this section we will be concerned with locating, identifying and tracking objects, though bear in mind that this is only one of many things vision does. For tasks like identifying discourse referents ("Get me the *cup*") or fetch-and-carry (locate the cup and grasp it) objects are the key ingredients.

Information obtainable by visual methods will often be incomplete (occlusions, backsides of objects, objects missed all together) or inaccurate

(uncertainties with regard to object identity or pose) and this uncertainty will have to be represented.

Visual knowledge can be characterised by the question "What is this?". This actually involves two unknowns: *what* and *this*. Vision must first segment the scene into things to reason about (*where* are objects?) and then gather information about those things (*what* are they?). Note that of course once we have learned object models, scene segmentation is no longer necessary and *where* and *what* are answered simultaneously. But for a self-extending, on-line learning system known object models can not be assumed a priori but rather are the results of learning activities. Furthermore search for known objects typically strongly benefits from knowledge *where* to concentrate search on.

So we have two questions to be answered regarding scene interpretation:

1. Where are objects?
2. What are the properties of objects (besides location)?

Where?

In the current architecture the first question "Where are objects?" or to be more precise "Where is a high likelihood of objects?" is addressed by a 3D attentional mechanism (plane pop-out), detailed in Section 6.1, which produces 3D spaces of interest (SOIs) that serve as object candidates. The question of "where?" is also addressed by tracking of learned object models detailed in Section 6.4.

Regarding the typology of incompleteness of Section 2 this is a **quantitative** incompleteness regarding the world **state**, with the nature of incompleteness being with respect to **variable value**. Once we have located a space of interest, the remaining incompleteness regards what is inside, as given in the next section.

What?

The second question "What are the properties of objects?" is addressed by (1) incremental learning and recognition of visual object properties as detailed in Section 8 about cross-modal beliefs and (2) learning and recognising identities detailed in Section 6.3, where we developed two approaches, one which outputs a distribution over object identities and views for a given image region (Section 6.3.1) and one which outputs single hypotheses of identity together with 6D object pose (Section 6.3.2).

Regarding the typology of incompleteness of Section 2 this constitutes a **qualitative** incompleteness over discrete world **state** (which of a number of known objects is it?), again with the nature of incompleteness being with respect to **variable value**. For the case of object recognition plus 6D location we additionally have **quantitative state** knowledge.

Observation Functions

Each method (detection, recognition, tracking) currently provides some form of confidence measure c, where each one lies in a different range (though often that range happens to be 0..1) and where each has a different actual meaning. To support Bayesian inference over the results of different methods we intend to transform these into actual probabilities by learning observation functions for confidence values.

Let us take (SIFT based) object recognition as an example. During the online recognition phase we can observe a confidence value (e.g. the relative number of matched SIFT features) which gives some information, though not always terribly accurate, about our confidence of having recognised the respective object. Now that value might consistently hover around 0.4 even for subjectively good results (note that half of the SIFT features will be on the backside of the object, some smaller features will often be lost due to scale changes). So the probability of a correct recognition is certainly not 0.4 (and why should it be - the confidence value is in no way intended to be a probability) but closer to 1.0. To obtain actual probabilities we therefore label training data with ground truth (e.g. bounding boxes around true object locations) and learn the probability of the confidence value c given true positives:

$$P(c \mid object = true) \tag{16}$$

This will typically produce a monomodal distribution (if not the confidence measure is chosen badly) to be expressed e.g. as a Gaussian. Learning these observation functions for each method then allows a consistent formulation of the probabilities of success or failure for each and thus supports probabilistic reasoning over all methods.

6.1 Bottom-up 3D Attention

Attention operators based on 2D image cues (such as colour, texture) are well known and discussed extensively in the vision literature but are not ideally suited for robotic applications. In such contexts it is the 3D structure of scene elements that makes them interesting or not. So our attention operator, plane pop-out, selects spaces of interest (SOIs) based on scene elements that pop out from supporting planes using 3D stereo data, which is detailed in Section 6.1.1 and 6.1.2 respectively. SOIs are then further refined by back-projection onto the 2D image followed by colour-based segmentation to produce what we call proto objects, detailed in 6.1.3.

6.1.1 Planes

Supporting planes are detected in the 3D point cloud reconstructed by stereo. To detect multiple planes, we apply an iterative RANSAC scheme,

where the consensus set of a detected plane is removed from the input points and search is repeated until eventually no further planes can be found. Hypothesis generation uses a bias where neighbouring points are selected in the sample set with a higher probability. Furthermore plane hypotheses that are not horizontal (note that we know the camera tilt angle) are rejected immediately. To estimate the confidence of a plane hypothesis i independent of the absolute number of inliers we use the point density

$$\rho_i = \frac{n_i}{A_i} \tag{17}$$

where n_i is the number of inliers of plane i and A_i is the area of the convex hull of the inliers projected onto the plane. High values of ρ indicate good plane hypotheses. But ρ is of course no actual probability, so we will learn an observation function

$$P(\rho \mid plane = true) \tag{18}$$

6.1.2 SOIs

Points not belonging to a plane are clustered and form popping-out spaces of interest (SOIs). SOIs are represented using the following data structure.

```
SOI
  boundingSphere
  foreGroundPoints
  backGroundPoints
  ambiguousPoints
```

The foreground points are those 3D points that stick out from the supporting plane and are enclosed by the bounding sphere. Background points are points that also fall inside the bounding sphere but are part of the supporting plane. These points capture the appearance of the local background around the objects. Finally ambiguous points form the boundary between supporting plane and segmented points and could not be labelled as foreground or background with certainty, i.e. they are near the RANSAC inlier threshold.

SOIs are tracked in the sequence of stereo frames by simply comparing their centres and sizes and a decision is made whether a new SOI was detected or an existing SOI was re-detected. As a next step we will replace this crisp decision with a confidence value c based on the overlap between SOI hypotheses. Using libelled training data we can then learn an observation function

$$P(c \mid soi = true) \tag{19}$$

6.1.3 Proto Objects

SOIs are object candidates which are segmented from their supporting planes. The quality of the segmentation depends on texture available for stereo reconstruction and often this segmentation is not very precise, especially concerning the above mentioned ambiguous points. To further move in the direction of proper objects the segmentation therefore is refined. To this end we back-project all 3D points into the image and perform 2D colour based segmentation, where colour samples for foreground and background are taken from the respective projected points. This leads to the definition of proto objects.

```
ProtoObject
  imagePatch
  segmentationMask
  foreGroundPoints
  SOI
```

The image patch contains the 2D image of the back-projected SOI and the segmentation mask the corresponding refined foreground/background labelling. The list of foreground points is the refined list of 3D points corresponding to the precise segmentation mask. Furthermore the proto object contains a reference to its source SOI. Proto objects now are entities already considered objects but with unknown attributes (apart from location). Filling in the attributes, answering the *what* question, is explained in the next section.

Once a proto object is detected, properties like colour and shape are filled in or object identity is recognised and the proto object is promoted to a visual object.

```
VisualObject
  // geometry
  geometryModel
  // identity
  identityLabels
  identityDistribution
  // location
  pose
  // properties
  propertyLabels
  propertyDistribution
  protoObject
```

A visual object is composed of several slots: geometry, identity and properties. Geometry encodes the 3D shape as a triangle mesh plus the 6D object pose, where pose can also represent the uncertainty of pose estimation. Object recognition will output various recognition hypotheses represented as a discrete probability density distribution over identities. Similarly recognition of properties outputs recognised properties as continuous probability density distribution.

6.2 Object shape: Detection

Detection of full 3D object shape is currently limited to basic geometric shapes such as cuboids (boxes) and cylinders. The approach detects groups of image edges corresponding to projections of basic shapes in non-degenerate views. Each grouping hypothesis has an associated significance value σ, which is derived from properties such as parallelism, closeness or completeness of Gestalt. Concretely we use the following for a cuboid (box):

$$\sigma_b = \frac{1}{6} \sum_{i=1..6} 1 - \frac{\delta_i}{\pi/2} \tag{20}$$

where δ_i are the angular differences between the opposing edge pairs of the three visible cuboid faces. These should be parallel (under orthographic projection) resulting in a significance value of 1 but are typically not due to effects of perspective as well as image noise. Similarly for cylinders:

$$\sigma_c = \frac{1}{2} \sum_{i=1..2} 1 - \frac{\delta_i}{\pi/2} \tag{21}$$

where we sum the angular differences between the occluding edges and the top and bottom ellipse major axes respectively.

These significance values are only meaningful however for ranking hypotheses of the same type of shape and are not simply comparable over types. So again we learn observation functions

$$P(\sigma_b \mid cuboid = true) \tag{22}$$

$$P(\sigma_c \mid cylinder = true) \tag{23}$$

6.3 Object identity: Recognition

Regarding object recognition we employ two approaches. Both are based on SIFT features. The first approach detailed in 6.3.1 uses a set of per-view bag-of-features object model and reasons about the identity of a given image (or image ROI) amongst several alternative hypotheses of object identity. The second approach detailed in 6.3.2 uses a 3D SIFT point cloud model and outputs only a single hypothesis of object identity together with full 6D pose.

6.3.1 Reasoning about object identity amongst alternatives

For a classifier to be used in a Bayesian network, it has to give the results of classification in a probabilistic form. For object recognition we use a general probabilistic model that is described in section 8.1.1. The model yields a probability distribution over all known classes, but also takes into account the case when the object being classified doesn't belong to any of the known

classes. This is described with the probability of the "unknown object" in the probability distribution. Here we describe the internal representation of a recognisable object and the calculation of the distance function $d(M_i, z)$ which is required by the probabilistic model.

Visual object model The object model M_i is represented with a set of views of the object from different viewpoints, $M_i = \{V_j^i\}_{j=1:N_i}$. A view V_j^i is described with viewpoint angles (ϕ, λ) and a set of features $\{F_k^v\}_{k=1:N_v}$ extracted from the image taken at this viewpoint. A feature $F_k^v = \{x_k, y_k, \sigma_k, \theta_k, \mathbf{X}_k\}$ is a SIFT feature described with its location in the image (x_k, y_k), scale σ_k, orientation θ_k and a 128-dimensional vector $\mathbf{X}_k$.

Similarly, an observation $z = \{F_k^z\}_{k=1:N_z}$ is represented with a set of SIFT features extracted from an image.

Probabilistic form Let $m \in \{m_k, m_u\}$ denote two possible events: (i) the observation came from an existing internal model m_k, and (ii) the observation came from an unknown model m_u. The general knowledge model is defined as a probability of observation z:

$$p(z) = p(z|m_k)p(m_k) + p(z|m_u)p(m_u). \tag{24}$$

The function $p(z|m_k)$ is the probability of explaining z given that z comes from one of the learnt models. The function $p(z|m_u)$ is the probability of z corresponding to the unknown model.

If we assume that the robot has learnt K separate alternative internal models $M = \{M_i\}_{i=1:K}$ from previous observations and we define the unknown model as M_0, the probability $p(z)$ can then be further decomposed:

$$p(z|m_k) = \sum_{i=1}^{K} p(z|M_i, m_k)p(M_i|m_k), \tag{25}$$

$$\begin{aligned} p(z) = p(m_k) \sum_{i=1}^{K} p(z|M_i, m_k)p(M_i|m_k) \\ + p(m_u)p(z|M_0, m_u)p(M_0|m_u). \end{aligned} \tag{26}$$

The general model requires a likelihood function $p(z|M_i)$ that gives the probability of observation z given the model M_i. We first define a distance function between the object model M_i and an observation z and use the distance function to define the likelihood function.

The likelihood function The object model representation is appearance based, so "appearance distance" is used to calculate the similarity between an observation z and a model M_i. When matching a view V_i^j of an object

model M_i to an observation z, the Euclidean distance between each pair of feature descriptors is calculated:

$$d_f(k,l) = d(\mathbf{X}_k^v, \mathbf{X}_l^z), k = 1 : N_v, l = 1 : N_z \tag{27}$$

where $\mathbf{X}_k^v$ is a SIFT descriptor from the view V_i^j and, $\mathbf{X}_l^z$ is a SIFT descriptor from the observation z. Let $l_{k,1}$ and $l_{k,2}$ be the indices of the shortest and second shortest distance between a descriptor from the model $\mathbf{X}_k^v$ and all descriptors from the observation z. In subsequent calculations we use only the shortest distances $d(k, l_{k,1})$ but only when the shortest two distances are far enough. Like Lowe [28] we use the threshold on the distance ratio to keep or reject feature matches:

$$d(k) = \begin{cases} d_f(k, l_{k,1}) & d_f(k, l_{k,1})/d_f(k, l_{k,2}) < 0.8, \\ -1 & otherwise. \end{cases} \tag{28}$$

The distances between a single view V_j^i and the observation z are converted to a view score D_j^i. The minimal distance d_0 is used to set an upper bound for the contribution of each matched SIFT feature to the final score. In practise d_0 can be either constant or estimated for each view separately in the learning phase.

$$D_j^i = \sum_{k,d(k)>0} \frac{1}{max(d_0, d(k))} \tag{29}$$

By taking the view with the highest score D_b^i where $b = \arg\max_j(D_j^i)$ we express the appearance distance $d(M_i, z)$ as

$$d(M_i, z) = \frac{1}{D_b^i} \tag{30}$$

The a priori probability of observing the i-th model, $p(M_i)$ can be set to $\frac{1}{K}$ if all known models are equally probable, or can be estimated from the number of times we have encountered the class M_i during learning. The a priori probability $p(m_u)$ that the observations will correspond to an unknown model can be set as in (31) and the PDF of the feature value z given the "unknown" model M_0 is set to a uniform distribution as in (32).

$$\begin{aligned} p(m_u) &= e^{\frac{-N*s}{\lambda_N}} \\ p(m_k) &= 1 - p(m_u). \end{aligned} \tag{31}$$

$$p(z|M_0) = \mathcal{U}(z). \tag{32}$$

The only distributions which remain to be estimated are the likelihood functions of each i-th class, i.e., $p(z|M_i, m_k)$. These are in fact probabilistic models that relate the likelihood of M_i being responsible for generating z, given the distance function value is $d(M_i, z)$. Formally,

$$p(z|M_i, m_k) = p(y_i|\Theta_i), \tag{33}$$

where $y_i = d(M_i, z)$ and Θ_i are the parameters of the probability density function corresponding to the i-th model.

The functional form of $p(y_i|\Theta_i)$ can be estimated experimentally. A common approach is to use common parameters for all distributions, thus $\Theta_i = \Theta$ for all i. Under the assumption that low distances $d(M_i, z)$ imply high probability of z corresponding to the model M_i and large distances imply low probability, a straightforward approach would be to define $p(\cdot|\Theta)$ as an exponential distribution, which requires estimating only a single parameter λ_d, e.g.,

$$p(y_i|\Theta) = {\lambda_d}^{-1} e^{-y_i/\lambda_d}. \tag{34}$$

This parameter can be estimated experimentally by obtaining a large number of typical values y_i for each model M_i and taking the $\hat{\lambda}_d$ that maximises the likelihood of these measurements under the model in (34).

Once the likelihood function $p(z|M_i)$ is defined, the a posteriori probability of a class M_i is calculated as

$$p(M_i|z) = \frac{p(z|M_i, m)p(M_i|m)}{\sum_{m \in m_u, m_k} \sum_{j=0}^{K} p(z|M_j, m)p(M_j|m)}, \tag{35}$$

where $m = m_k$ for $i \in [1, K]$ and $m = m_u$ for $i = 0$. The index of the most probable object model $M_{\hat{i}}$ given the observation z is

$$\hat{i} = \arg\max_i(p(M_i|z)). \tag{36}$$

6.3.2 Locating 3D object instances

While the approach detailed in the previous section reasons about object identity of a given image region and outputs a distribution over object identities, the following approach outputs a single object identity together with 6D pose. The approach is based on SIFT features, which are mapped onto the 3D geometric surface of an object during a learning phase. SIFT descriptors are then used to build a codebook, where descriptors are clustered using an incremental mean-shift procedure and each 3D location on the object surface is assigned to the according codebook entry.

In the recognition phase SIFT features are detected in the current image and matched with the codebook. According to the codebook entry each

matched feature has several corresponding 3D model locations. To robustly estimate the 6D object pose we use the *OpenCV* pose estimation procedure in a RANSAC scheme with a probabilistic termination criterion. Given an acceptable failure rate η_0, i.e. the accepted probability that no valid sample set could be found we can derive the number of iterations k necessary to achieve a desired detection probability $(1 - \eta_0)$ using an estimate of the inlier ratio $\hat{\epsilon}$, which is taken to be the inlier ratio of the best hypothesis so far:

while $\eta = (1 - \hat{\epsilon}^m)^k \leq \eta_0$) **do**
...
end while

with m the size of the minimum sample set. So the number of RANSAC iterations is adapted to the difficulty of the current situation and accordingly easy cases quickly converge.

We define the confidence of detection

$$c = \frac{n_{inlier}}{n_{detected}} \tag{37}$$

as the ratio between the matched interest points n_{inlier} and the number of detected interest points $n_{detected}$ located within the object boundary projected to the current image. This provides a good estimate independent of the total number of features in the model and independent of current scale.

This confidence is no actual probability however so we will again learn an observation function

$$P(c \mid object = true) \tag{38}$$

6.4 Object location: Tracking

The following section deals with tracking of learned object models. Models are given as 3D shape models (i.e. triangulated meshes) together with surface texture and we estimate the probability distribution over full 6D object pose.

To allow estimation of non-parametric and possibly multi-model PDFs, our approach uses Monte Carlo particle filtering. Concretely we model a moving object as a dynamic system with state $\mathbf{x}$, system function $f(.)$, system noise w_k, observation function $h(.)$ and observation noise v_k

$$x_{k+1} = f(x_k) + w_k \tag{39}$$
$$y_k = h(x_k) + v_k \tag{40}$$

where $\mathbf{x}$ is 6D object pose, $f(.)$ is a linear motion model and $h(.)$ represents the pinhole camera projection. w_k and v_k are zero mean Gaussian noise.

The PDF over 6D object pose $\mathbf{x}$ is approximated using a set of N samples, or particles $\mathbf{x}_i$ and associated importance weights w_i.

$$\hat{\mathbf{x}} = \sum_{i}^{N} w_i \delta(\mathbf{x_i}) \tag{41}$$

where $\delta(\mathbf{x_i})$ denotes the Dirac delta mass located at $\mathbf{x_i}$. In each tracking step k, weights are evaluated by projecting the object into the image using the respective pose and drawing contours as well as surface texture edges and subsequently comparing these with edges extracted from the camera image. The samples $\mathbf{x}_i$ are resampled according to the importance weights of the prior distribution and the linear motion model as well as Gaussian system noise w_k are applied, where higher noise levels allow tracking of faster moving objects (by allowing particles to "explore a larger search space").

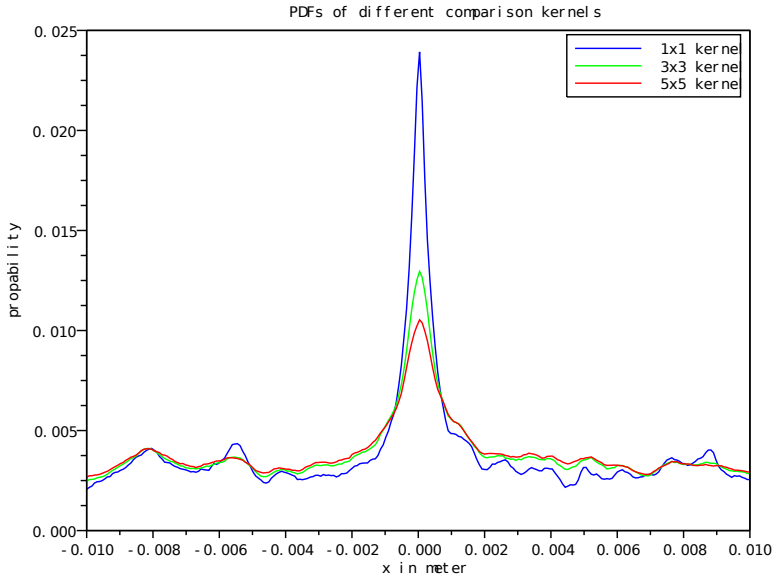


Figure 10: Probability density function of a translational degree of freedom using different levels of observation noise (see text for details)

Observation noise v_k corresponds to the amount of dilation applied to image edges prior to matching with projected model edges. Fig. 6.4 presents the probability density function of one of the 6 DOF of the tracking state $\mathbf{x}$ and shows how higher levels of observation noise tend to smooth estimated PDFs.

6.5 Conclusion

Visual knowledge as detailed in the previous sections is always related to (some parts of) the world **state**. This is not surprising as vision on its own

as we use it now is concerned with perceiving the world rather than explaining it (cause, effect knowledge). By combining visual information later on with information from action we talk about causes and effects. Concretely, in this section we were interested in object locations and identities. The former constitutes **quantitative** knowledge (expressed e.g. as continuous probability distributions in the tracker) while the latter constitutes quantitative knowledge about qualitative entities (expressed e.g. as discrete probability distributions in case of object recognition). Incompleteness of knowledge then relates to **variable values** (e.g. the 6D pose of an object).

Not all visual methods can provide genuine probabilities of success or failure, but do typically provide some measure of confidence. To support consistent probabilistic reasoning over the results of different methods we will learn observation functions for these confidence values.

7 Planning: how beliefs change under action

Planning is concerned with the synthesis of action strategies that bring about the achievement of objectives. Planning procedures derive such strategies given a model of the environment. That model describes the starting conditions of the robot, along with the actions that it can execute, and the objectives that it seeks to achieve. Essentially, the model corresponds to a detailed description of an agent's knowledge about its environment.

In the sequel we formally describe how we represent models of the environment for the purposes of planning. We give a description of what we consider to be a propositional planning problem where action effects are deterministic. Equivalent to the STRIPS planning model [16], this formalism corresponds to a coarse-grained solution to the: (1) *frame* problem —i.e., modelling what does not change in the environment when the robot acts in the environment; (2) its dual, the *ramification* problem —i.e., modelling what changes as the robot acts; and (3) the *qualification* problem —i.e., modelling what preconditions must be met in order for an operation to be performed. Following our exposition of propositional planning we motivate a relational formalism, called PDDL,[4] that we use to represent planning problems. Making all the above ideas concrete, we describe a simple version of the CogX *dora* scenario using PDDL.

Of course, that is not sufficient to address the ongoing concerns we have in CogX about beliefs, and in particular representations of a machine's belief about its environment. In particular, we cannot deal with situations where the robot believes it could be in any one of a number of states, that it believes an action can have one of a number of effects, and where it believes that inference to the *true* world-state is based on a concrete probabilistic scheme

[4]Planning Domain Definition Language.

for gathering perceptual evidence. In other words, using our deterministic propositional formalism we cannot address:

- Uncertainty in state —e.g., The cornflakes could be in the kitchen, or in the lounge room.
- Uncertainty in acting —e.g., When *dora* opens a door, it could be locked and therefore might not open. *A priori* (before acting) *dora* does not know the outcome of trying to open the door.
- Uncertainty in perception —e.g., Just because *dora* perceives cornflakes, how should that effect her belief that cornflakes are really in her visual field?

In order to address those requirements, we appeal to much more powerful representational mechanisms, in particular the MDP[5] for modelling uncertainty in action outcomes, and the POMDP[6] for modelling uncertainty in state and perception. In the sequel we review those in turn, and then motivate a more expressive variant of PDDL, called DTPDDL,[7] a relational formalism that is rich enough to describe stochastic domains with partial observability. The details of that language are given in Appendix A. We sketch the basic features of that language and our modelling formalism by giving a richer example of a *dora* scenario that exhibits state-uncertainty and partial observability.

Finally, CogX requirements motivate an explicit representation of (knowledge) gaps . This is of particular importance when enumerations of possible states or action outcomes are not available – e. g., when the name of a human is unknown and the robot plans to discover it. For our approaches to planning in multiagent environments —e. g., Human-Robot Interaction— we therefore use a planning representation that explicitly models knowledge and how it can be affected by epistemic operator effects, MAPL[8]

Making those ideas concrete, we describe a simple version of the CogX *dora* scenario using MAPL.

7.1 Classical Planning

Here we describe deterministic propositional planning formally. In the following section we motivate PDDL for CogX; That is the *de facto* planning domain definition language for propositional planning. Subsequently we

[5]Markov Decision Process.
[6]Partially Observable Markov Decision Process.
[7]Decision-Theoretic PDDL.
[8]Multiagent Planning Language.

make these ideas concrete with an example, before developing representational extensions to this material that we use in CogX.

A propositional planning problem is given in terms of a finite set of objects $\mathcal{O}$, first-order STRIPS-like planning operators of the form:

$$\langle o, pre(o), add(o), del(o) \rangle$$

and predicates Π. Here, o is an expression of the form $\mathcal{O}(x_1, \ldots, x_n)$ where $\mathcal{O}$ is an operator name and x_i are variable symbols, $pre(o)$ are the operator preconditions, $add(o)$ are the add effects, and $del(o)$ the delete effects. By grounding Π over $\mathcal{O}$ we obtain the set of propositions P that characterise problem states. For example, suppose we have a robot called *dora* who can be in the *Library* or the *Kitchen*. Then we can have a binary predicate `In`, one grounding of which is the proposition $\texttt{In}(\texttt{Dora}, \texttt{Library})$. A planning problem is posed in terms of a starting state $s_0 \subseteq P$, a goal $\mathcal{G} \subseteq P$, and a small set of domain operators.

An action a is a ground operator having a set of ground preconditions $pre(a)$, add effects $add(a)$, and delete effects $del(a)$. The contents of each of those sets are made up of elements from P. An action a can be executed at a state $s \subseteq P$ when $pre(a) \subseteq s$. We denote $\mathcal{A}(s)$ the set of actions that can be executed at state s. When $a \in \mathcal{A}(s)$ is executed at s the resultant state is $(s \cup add(a)) \backslash del(a)$. Actions cannot both add and delete the same proposition – i.e., $add(a) \cap del(a) \equiv \emptyset$.

A state s is a *goal state* iff $\mathcal{G} \subseteq s$. A plan is a prescription of non-conflicting actions to each of n time steps. We say that a plan solves a planning problem when executing all the actions at each step starting from s_o achieves a goal state. A plan is optimal iff no other plan can achieve the goal in a shorter number of time steps. The planning problem just described is PSPACE-complete in general, and NP-Complete under fairly reasonable restrictions to plan length.

7.2 PDDL

We have just describe planning in a propositional sense, and thus far only alluded to the fact that in practice we employ a more sophisticated and general formalism. This is required because in robotics applications, collections of problems usually exhibit a strong relational structure and are therefore best represented using first-order languages supporting the declaration of objects and relations over them as well as the use of quantification over objects [30]. The 3rd Planning Domain Definition Language (PDDL3.0 – pronounced "pea-diddle") is the language of choice of the planning community [29, 18, 14]. This language and its predecessors have been developed for and adopted at the International Planning Competitions since 1998.

A PDDL-based language forms the basis of domain and problem description in the planning subarchitecture of CogX. The details of that grammar

are given in Appendix A. For the purposes of this document, we shall elaborate on the ideas of the previous section and Appendix A by given an example from a familiar CogX scenario.

7.2.1 Example: Deterministic Representations

Suppose we have a robot, *dora*, that can be located in one of four `places`:

1. Kitchen
2. Library
3. Office
4. Hall

We suppose all of those places are connected via the `hall`. For example, to traverse from the `kitchen` to the `library`, Dora must pass through the `hall`. Now, suppose that *dora* has knowledge about the appearance of 4 types of object:

1. Bookshelves
2. Desktops
3. Chefs
4. Cornflakes

Here, `cornflakes` are a special type of object that she can interact with – For our purposes, the object `cornflakes` is a member of a *type* called `widget`. We typically have that the starting environment is characterised by the set of facts that are true. In PDDL, *dora's* starting environment is given as follows.

First, we describe a `problem`, called "dora_the_explorer_problem1", that is an instance of a `domain`, called "dora_the_explorer". We then describe the objects that occur for this problem, along with their type.

```
(define (problem dora_the_explorer_problem1)
  (:domain dora_the_explorer)
  (:objects
   Hall R1 R2 R3 - place
   Library Kitchen Office Hall-Label - label
   Cornflakes - widget
   Bookshelf Desktop Chef - feature
   )
```

Following this, we describe the starting state, that corresponds to the set of facts that are true in the initial configuration of *dora*.

```
  (:init (connected Hall R1) (connected R1 Hall)
         (connected Hall R2) (connected R2 Hall)
         (connected Hall R3) (connected R3 Hall)

         (assign (labelled R1) Library)
         (assign (labelled R2) Kitchen)
         (assign (labelled R3) Office)

         (widget-location Cornflakes R2)
         (featured-at Bookshelf R1)
         (featured-at Chef R2)
         (featured-at Desktop R3)

         (assign (located) Hall)
  )
```

Above, we have put the `cornflakes` in the `kitchen` along with the `chef`, and then a `bookshelf` in the `library`, and a `desktop` in the `office`. We have also reflected the connectivity between the rooms in *dora's* environment, and placed her in the `hall` initially.

Lastly, we can describe what objectives/goals *dora* should act to achieve. In this case, we suppose she wants to be in the same room as where the `cornflakes` are located.

```
  (:goal (= (located) R2))
)
```

Recapping, we have now described *dora's* starting configuration, along with the goal configuration. We have also described the objects in her world, along with their respective types. In order to complete the model, it is left to describe a type hierarchy (over object types), along with a description of the relations that exists between objects in *dora's* world, and the actions that can affect changes in her world. These aspects of the model form part of what we call the *domain* (resp. *problem*) description. In this case, the domain is what we referred to earlier as "*dora_the_explorer*". The prefix to a domain description gives an identifying string, along with a list of classes of descriptive elements we required, called the "*:requirements* string".

```
(define (domain dora_the_explorer)

  (:requirements
   :typing
   :strips
   :equality
   :fluents )
```

Here, inclusion of *:typing* means that objects in our domain are typed and that the domain description shall contain a type hierarchy. The string ":strips" means that we want to use PDDLs syntactic elements for describing propositional planning problems. String ":equality" gives us access to

equality – i.e., the "(= ...)" predicate, that can take 2 *object* arguments of any *type*, and evaluates to true if both arguments are equal. Finally, ":fluents" allows us to use functional elements in our model – e.g., "(assign (location) Hall)" in our starting state description.

The *type* hierarchy for *dora* occurs as follows:

```
(:types
 place label widget feature - object
 model - (either widget feature)
 )
```

Above, we suppose `widget`s, `feature`s, and `place`s and their associated `label`s are all *objects*. We also introduce a `model` *type*, instances of which are either `widget`s or `feature`s.

Following a description of the types in the domain description, we can give the relations and concepts that exists within the world. In particular, we give the *:predicates* and state functions *:s-functions*. The symbols we declare below were all used to describe *dora's* starting configuration earlier.

```
(:predicates

 (connected ?p1 - place ?p2 - place)
 (widget-location ?o - widget ?p - place)
 (featured-at ?model - feature ?location - place)
)


(:s-functions

 (labelled ?r - place) - label

 (located) - place
)
```

Above, symbols have a ? prefix if they correspond to a variable name. Strings "- `type-name`" give the *types* of the proceeding variable symbols. For example, above predicate `connected` is binary, taking arguments of type `place`.

Lastly, in the domain description we give the action-physics of *dora's* world. In this case, we describe how *dora* can move between connected places using the action schema that follows.

```
(:action move-to-connected-place
         :parameters (?to - place ?from - place)
         :precondition
         (and
          (= (located) ?from)
          (connected ?from ?to)
          )
         :effect
         (and
          (assign (located) ?to)
          )
         )
 )
```

Given our starting configuration, we have that a ground instance of this action is:

$$\texttt{move-to-connected-place}(\texttt{Hall}, \texttt{Kitchen})$$

This ground action is only executable when *dora* is `located` in the `hall`. Recall, from our starting configuration, that the `hall` is `connected` to the `kitchen`. The effect of executing this action is to have the function that tracks *dora's* location, i.e., `located`, changed to reflect that *dora* moved from the `hall` to the `kitchen`.

7.3 Decision-Theoretic Planning

Here we examine the representations of beliefs in planning. First we examine the case that there is quantified uncertainty about the effects of actions. Then, we move on to the case where the state of the world is only partially observable. In that case we have quantified uncertainty about the state of the environment, and also the robot must gather evidence, via perception, about the true environment – i.e., the true state of the world might not be observable(/knowable).

7.3.1 Markov Decision Processes

The following notes describe the MDP formalism developed in [21]. A much more detailed coverage of the material of this section can be found in [30] and [6].

A Markov decision process (MDP) is defined in terms of a four-tuple $\langle \mathcal{S}, \mathcal{A}, \mathsf{Pr}, \mathsf{R} \rangle$. $\mathcal{S}$ is a finite set of states. We write $\mathcal{S}^*$ for the set of finite sequences of states over $\mathcal{S}$. Also, Γ is a member of $\mathcal{S}^*$, and where i is a natural number, Γ_i is the state at index i in Γ, and $\Gamma(i)$ is the prefix $\langle \Gamma_0, \ldots, \Gamma_i \rangle \in \mathcal{S}^*$ of Γ. $\mathcal{A}$ is a finite set of actions. Where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}$ then $\mathsf{Pr}(s, a, s')$ is the probability of a transition from state s to s' given action a is executed at state s. Naturally then, for any s and a we have the following constraint:

$$\sum_{s' \in \mathcal{S}} \mathsf{Pr}(s, a, s') = 1$$

Also present is a bounded real-valued reward function $\mathsf{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Re$. R is bounded if there is a positive constant c so that for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$, $|\mathsf{R}(s, a)| < c$.

The solution to an MDP is called a policy, which is a prescription of how actions are chosen at a history. Popular classes of policy include:

- A stationary (deterministic) policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ as a total function mapping states to actions,

- A stochastic memoryless policy $\pi : \mathcal{S} \to P_{\mathcal{A}}$ so that $\pi_a(s)$ is the probability that we execute a given we are in s, and
- A policy as a deterministic function of the state history $\pi : \mathcal{S}^* \to \mathcal{A}$.[9]

We typically suppose the robot will act in the MDP forever, thus an optimal policy is one that maximises the discounted cumulative reward over an infinite horizon. Writing $\mathsf{R}^{\pi}_{\mathsf{seq}}$ for the reward function that accumulates R over some finite prefix Γ_i generated according to stationary policy π, the value of a policy is as follows.

$$V_{\pi}(s) = \lim_{n\to\infty} \mathsf{E}\left[\sum_{i=0}^{n} \beta^i \mathsf{R}^{\pi}_{\mathsf{seq}}(\Gamma_i) \mid \pi, \Gamma_0 = s\right] \tag{42}$$

Where π is stationary and deterministic, as a matter of convenience we often prefer to express the value function in terms of a Bellman equation:

$$V_{\pi}(s) = R(s, \pi(s)) + \beta \sum_{s' \in \mathcal{S}} \mathsf{Pr}(s, \pi(s), s') V_{\pi}(s') \tag{43}$$

In (Eqns 42 and 43) $0 < \beta < 1$ is a discount factor expressing the relative importance of imminent versus distant rewards. A policy π^* is optimal if, for all π of any type, we have $\forall s \in \mathcal{S},\ V_{\pi^*}(s) \geq V_{\pi}(s)$. There is always an optimal policy π^* that is stationary – I.e., $\pi^* : \mathcal{S} \to \mathcal{A}$. Thus, the solution to a discounted infinite horizon fully-observable decision-theoretic planning problem is a stationary policy.

Given MDP $\langle \mathcal{S}, \mathcal{A}, \mathsf{Pr}, \mathsf{R} \rangle$, we seek an ϵ-optimal policy π^* for discount factor $0 < \beta < 1$.

Definition 1. *The Bellman operator T is a mapping defined so that $T(V) = V'$ if for every state $s \in \mathcal{S}$*

$$V'(s) = \max_{a \in \mathcal{A}} [\mathsf{R}(s, a) + \beta \sum_{s' \in \mathcal{S}} \mathsf{Pr}(s, a, s') V(s')]$$

□

We have that the value function associated with an optimal policy is the unique fixed-point solution to this set of equations. I.e.,

$$V_{\pi^*} = T(V_{\pi^*}) \tag{44}$$

Existence and uniqueness follow from the fact that T is an infinity-norm contraction. Indeed, where $\|X\|_{\infty} = \max_{x \in X} |x|$ —writing $|x|$ for the absolute value of x— we have that for any two value functions V_1 and V_2

[9] We will see that the latter are not very interesting for the case of MDPs, however are necessary for acting optimally in the n-horizon POMDP problem to be discussed in a moment.

$$\|T(V_1) - T(V_2)\|_\infty \leq \beta \|V_1 - V_2\|_\infty \tag{45}$$

From Banach's fixed-point theorem, we have the following for any π^*.

$$\lim_{n\to\infty} T^n(V) = V_{\pi^*} \tag{46}$$

For any V, acting greedily according to $\lim_{n\to\infty} T^n(V)$ corresponds to acting optimally.

7.3.2 Partially Observable Markov Decision Processes

The current *de facto* model for probabilistic decision-theoretic planning with partial observability is the POMDP. For our purposes, a POMDP is a six-tuple $\langle \mathcal{S}, \mathcal{A}, \mathsf{Pr}, \mathsf{R}, \mathbb{O}, v \rangle$. Here, $\mathcal{S}$, $\mathcal{A}$, Pr, and R are states, actions, state-transition function, and reward function, respectively – They provide an MDP-based specification of the underlying world state, dynamics, and reward. $\mathbb{O}$ is a set of observations. For each $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, an observation $o \in \mathbb{O}$ is generated independently according to some probability distribution $v(s, a)$. We denote $v_o(s, a)$ the probability of getting observation o in state s. For s and a we have the following constraint:

$$\sum_{o\in\mathbb{O}} v_o(s, a) = 1$$

The optimal solution to a finite-horizon POMDP problem can be expressed as a policy $\mu : \mathbb{O}^* \to P_{\mathcal{A}}$ where $\mu_a(o_0, .., o_t)$ is the probability that we execute action a given observation history $o_0, .., o_t$.[10] A finite-state controller (FSC) is a more useful policy representation mechanism in the case that the robot has ongoing interactions with the environment modelled by the POMDP at hand. This is a three-tuple $\langle \mathcal{N}, \psi, \eta \rangle$ where: $n \in \mathcal{N}$ is a set of nodes, $\psi_n(a) = P(a|n)$, and $\eta_n(a, o, n') = P(n'|n, a, o)$. The value of state s at node n of the FSC for a given POMDP is:

$$V_n(s) = \sum_{a\in\mathcal{A}} \psi_n(a)\mathsf{R}(s, a) + \beta \sum_{a,o,s',n'} \eta_n(a, o, n')\mathsf{Pr}(s, a, s')v_o(s', a)V_{n'}(s') \tag{47}$$

If b is a POMDP belief state – i.e, $b(s)$ gives the probability that the robot is in state s – then the value of b according to the FSC is:

$$V_{\mathsf{FSC}}(b) = \max_{n\in\mathcal{N}} \sum_{s\in\mathbb{S}} b(s)V_n(s) \tag{48}$$

[10] Such a policy can oftentimes be compactly represented as a tree or algebraic decision diagram (ADD).

7.4 DTPDDL

DTPDDL extends PDDL in four key respects. First, the effects of actions can be stochastic, and therefore the language facilitates the specification of action effects with quantified uncertainty. Second, a DTPDDL domain description explicitly labels predicate and function symbols as being about the underlying state (unobservable) or about the agents perceptions (observable). Thirdly, we introduce syntax for describing observation schemata. This is used to specify how action executions in states generate observations. Finally, the fragment of PDDL for problem definitions does not support specification of a problem start state distribution – i.e., full observability is assumed, and hence the starting state is supposed to be fully observable. For DTPDDL we introduce syntax for compactly specifying the starting state distribution.

7.4.1 Example: Representations of Quantified Uncertainty

Here we extend the *dora* scenario given earlier by adding a Bayesian-style personal belief about the world, in the form of a starting-state distribution. We also make the underlying task much more complicated, by adding partial observability. Here *dora* is supposed to act in order that she can make a solid commitment about the *true* location of the `cornflakes`. If she makes the correct commitment in this regard, then she is rewarded strongly, and otherwise she is punished terribly. Consequently, our *dora* has an interest in disambiguating via perception to find the underlying world state, especially where that state information pertains to the true location of `cornflakes`.

Suppose that *dora* can be a little bit forgetful, and thus keeps a photograph of the `models` that occur in the world, including a photo of `cornflakes`. Photos that she can refer to while searching occur in slots that she keeps in a pouch around her tummy. Unfortunately, there is usually a very limited number of slots, and therefore she has to be very careful what photos she keeps in her slots when deliberating towards a particular disambiguation objective.

For our new example, the problem description prefix defines the available objects as follows:

```
(define (problem dora_the_explorer_POMDP)
  (:domain dora_the_explorer_POMDP)
  (:objects
   Hall R1 R2 R3 - place
   Library Kitchen Office Hall-Label - label
   Cornflakes - widget
```

```
Bookshelf Desktop Chef - feature

S1 - model-slot

)
```

Here, we suppose *dora* has a single slot `S1`. The initial state is significantly different from the deterministic case, because now we must detail the set of states *dora* might be in, and the probability that she is in any particular state – i.e., describe her Bayesian belief-state about her environment. Thus, we have a few deterministic state elements:

```
(:init (connected Hall R1) (connected R1 Hall)
       (connected Hall R2) (connected R2 Hall)
       (connected Hall R3) (connected R3 Hall)

       (= (foregrounded-model S1) Empty)

       (= (reward) 0)

       (deletable S1)

       (assign (located) Hall)

       (labelled Hall Hall-Label)
```

And then, we have to describe the probability that : (1) `place`s have particular `label`s, and (2) that `place`s feature particular `model`s. A partial description of the starting state distribution follows:

```
(probabilistic 1/8 (and (assign (labelled R1) Library)
 (assign (labelled R2) Kitchen)
 (assign (labelled R3) Office)
 (probabilistic 0.9 (featured-at Bookshelf R1) )
 (probabilistic 0.3 (featured-at Bookshelf R3) )
 (probabilistic 0.8 (featured-at Chef R2)
  0.1 (featured-at Chef R1)
  0.1 (featured-at Chef R3) )
 (probabilistic 0.9 (featured-at Desktop R3) )
  (probabilistic 0.3 (featured-at Desktop R1) )
  (probabilistic 0.1 (featured-at Desktop R2) )
  (probabilistic 0.8 (widget-location Cornflakes R2)
   0.2 (widget-location Cornflakes R3) )
 )


 1/8 (and (assign (labelled R1) Library)
 (assign (labelled R2) Office)
 (assign (labelled R3) Kitchen)
 (probabilistic 0.9 (featured-at Bookshelf R1) )
 (probabilistic 0.3 (featured-at Bookshelf R2) )
 (probabilistic 0.8 (featured-at Chef R3)
  0.1 (featured-at Chef R1)
  0.1 (featured-at Chef R2) )
 (probabilistic 0.9 (featured-at Desktop R2) )
  (probabilistic 0.3 (featured-at Desktop R1) )
  (probabilistic 0.1 (featured-at Desktop R3) )
  (probabilistic 0.8 (widget-location Cornflakes R3)
   0.2 (widget-location Cornflakes R2) )
```

```
)
.....
```

Above, we have that there is a 1/8 chance that `place R2` is a `kitchen`, and a 1/8 that it is an `office`. In the first case it is very likely (90% chance) that the `library` features a `bookshelf`, and so on...

We also should describe, for the starting configuration, the probability that *dora* observes a `feature` at a `place` supposing that `place` has a particular `label`. In other words, we must give a model of her perception. That model determines how *dora* can change her belief about the labels of places, and the locations of features – i.e., by placing photos in her slot and then exploring the world. In the starting state description we have:

```
  (assign (probability__observe_feature_at_place_with_label
           __if_true R1 Bookshelf Library) 0.9 )
  (assign (probability__observe_feature_at_place_with_label
           __if_true R1 Bookshelf Kitchen) 0.1 )
  (assign (probability__observe_feature_at_place_with_label
           __if_true R1 Bookshelf Office) 0.3 )
.............
  (assign (probability__observe_feature_at_place_with_label
           __if_feature_false R3 Chef Library) 0.2 )
  (assign (probability__observe_feature_at_place_with_label
           __if_feature_false R3 Chef Kitchen) 0.3 )
  (assign (probability__observe_feature_at_place_with_label
           __if_feature_false R3 Chef Office) 0.2 )
.............
```

From the first three *assignments* above, we have that if the true `label` of `R1` is `library`, then *dora* shall observe a `bookshelf` at `R1` 90% of the time when she explores that `place`, provided a `bookshelf` is indeed present. From the second three *assignments*, *dora* has a %20 chance of observing a `chef` in `R3` labelled `library` given the `chef` is not actually present – i.e., She cannot trust her perception of the `chef`. The purpose of these observational probabilities shall be further clarified when we give the observation schema to describe *dora's* perceptual model for our POMDP setting.

We also have to describe the probability of observing a `widget` in a `place` given it has a particular `label`. Here, the "__T" suffix indicates that we are expressing the probability when the `widget` is present, and "__F" is for the case that it is not present.

```
  (assign (probability__observe_widget_model_at_label__T Library
                                                         Cornflakes)
          .7)

  (assign (probability__observe_widget_model_at_label__T Kitchen
                                                         Cornflakes)
          .7)
  (assign (probability__observe_widget_model_at_label__T Office
                                                         Cornflakes)
          .7)

  (assign (probability__observe_widget_model_at_label__F Library
                                                         Cornflakes)
```

```
        .1)
  (assign (probability__observe_widget_model_at_label__F Kitchen
                                                         Cornflakes)
        .1)
  (assign (probability__observe_widget_model_at_label__F Office
                                                         Cornflakes)
        .1)
```

Finally, we have to give *dora's* overall objective. In this case, we have:

```
  (:metric maximize (reward))
)
```

The above specifies that *dora* should try to maximise her reward. We shall see how she might go about that in a moment, when we describe how *dora* can act and perceive with stochastic actions and partial observability.

The domain description prefix is more detailed for our reworked example, as our "*:requirements*" string must specify that we have stochastic actions with probabilistic effects, and that we will give observation schemata – i.e., use "*:observe*" schemata to model *dora's* perception.

```
(define (domain dora_the_explorer_POMDP)

  (:requirements
   :typing
   :strips
   :equality
   :fluents

   :probabilistic-effects

   :universal-effects
   :conditional-effects

   :partial-observability
  )
```

The description of `types` in *dora's* environment is only altered slightly from our deterministic domain description. These modifications take into account *dora's* photo pouch (cf. `model-slot`).

```
  (:types
   place label widget feature  model-slot - object
   model - (either widget feature)
   )
```

In describing the predicates of *dora's* world, we add one binary predicate *absolute_belief__widget_location* to those given for the deterministic scenario. That expresses the commitments *dora* has made to the locations of widgets. In particular, we have:

```
  (:predicates
   (explored ?p - place)
   (connected ?p1 - place ?p2 - place)
   (widget-location ?o - widget ?p - place)
   (featured-at ?model - feature ?location - place)

   ;; What commitments has Dora made to the location ?loc of widget ?w
   (absolute_belief__widget_location ?w - widget ?loc - place)
   )
```

We must also give the functions that we use to encapsulate *dora's* environment.

```
(:s-functions
 (labelled ?r - place) - label     ;; The label of a place
 (located) - place                 ;; Where Dora is located
 (reward) - double                 ;; The reward Dora has accumulated

 ;; What model/photo is Dora storing in slot ?s?
 (foregrounded-model ?s - model-slot ) - model


 ;; Probability Dora sees ?m, and it is there.
 (probability__observe_feature_at_place_with_label
  __if_true
   ?loc - place
   ?m - feature
   ?l - label
 ) - double

 ;; Probability Dora sees ?m, but it is not there.
 (probability__observe_feature_at_place_with_label
  __if_feature_false
   ?loc - place
   ?m - feature
   ?l - label
 ) - double

 ;; Probability Dora sees ?w, and it is there.
 (probability__observe_widget_model_at_label
  __T
   ?l - label
   ?w - widget
 ) - double

 ;; Probability Dora sees ?w, but it is not there.
 (probability__observe_widget_model_at_label
  __F
   ?l - label
   ?w - widget
 ) - double


)
```

We also suppose there is a constant in this domain that we use to model the notion that *dora's* `model-slot` is `Empty` – i.e.,

```
(:constants

 Empty - feature


)
```

For example, the condition that *dora* has no photos in `model-slot S1` is expressed in DTPDDL as:[11]

[11]The constant `Empty` is an artifact of the fact that we cannot specify partial *object valued fluents* —i.e., functions with finite range— in PDDL.

$$(= \ (\texttt{foregrounded-model S1}) \ \texttt{Empty})$$

Whereas in our deterministic example *dora* was able to perceive the state of her environment exactly, in this example she only has access to atoms of observation called *percepts*. That is, *dora* cannot observe elements in "*s-functions*" and "*:predicates*" description elements unless they are known with certainty according to her belief-state distribution. However, she can observe her "observation" state exactly. The propositions that make up the observation state are given in a "*:percepts*" description fragment. In particular, we suppose that *dora* is able to instantaneously perceive `model`s in `place`s.

```
(:percepts
 (observed_model_at_place ?n - place ?m - model)
 )
```

For example, we write:

$$(\textit{observed\_model\_at\_place} \ \texttt{Kitchen Cornflakes}) \tag{49}$$

to express the observational fact that *dora* has perceived `Cornflakes` in the `Kitchen`.

We can now describe the action physics for our *dora* scenario. In the first place, *dora* can move between connected places, and also explore a place she is in. We suppose that when exploration is invoked that *dora's* visual abilities are employed to discern whether a photograph in her slot occurs in the place she is exploring. The details of that perceptive inference are modelled later in terms of an "*:observe*" schema.

```
(:action move-to-connected-place
         :parameters (?to - place ?from - place)
         :precondition
         (and
          (= (located) ?from)
          (connected ?from ?to)
          )
         :effect
         (and
          (assign (located) ?to)
          )
         )
 )

(:action explore-place
         :parameters (?loc - place)
         :precondition (and
                        (= (located) ?loc)
                        )
         :effect (and (explored ?loc))
         )
```

We also give *dora* the ability to focus or change the photographs she keeps in her slot. We suppose she can "*foreground*" a `model` by retrieving it from her pouch and placing it in a free `model-slot`. Moreover, she can return a "*foregrounded*" model to her pouch from an occupied `model-slot`, thus freeing that slot for further use. The latter we suppose is a "*backgrounding*" action. Those two actions, *foregrounding* and *backgrounding*, are represented as follows.

```
(:action foreground_model
  :parameters (?m - model ?s - model-slot)
  :precondition (and
   ;; Can only foreground a model to an empty model-slot.
   (= (foregrounded-model ?s) Empty)
   ;; Test that Dora has not already foregrounded the model
   (forall (?s2 - model-slot)
    (not (= (foregrounded-model ?s2) ?m)))
   )
  :effect (assign (foregrounded-model ?s) ?m)
)

(:action background_model
  :parameters (?s - model-slot)
  :precondition ()
  :effect (assign (foregrounded-model ?s) Empty)
)
```

Another action we allow *dora* to perform is that of making a commitment to the location of a widget. Here, we suppose she achieves a large reward, $1000, when she commits to the correct location, and is punished with a $500 fine when she makes an incorrect commitment.

```
(:action commit__widget_location
         :parameters (?w - widget ?loc - place)

         :precondition (forall (?loc2 - place)
              (not (absolute_belief__widget_location ?w ?loc2)))

         :effect (and (absolute_belief__widget_location ?w ?loc)
                      (when (widget-location ?w ?loc)
                        (increase (reward) 1000.0))
                      (when (not (widget-location ?w ?loc))
                        (decrease (reward) 500.0)))
         )
```

We now provide the details of *dora's* perceptual model. This is achieved by giving "*:observe*" schemata whose preconditions are over state predicates (and functions), and whose effects are over observational predicates. Such schemata also contain an "*:execution*" precondition, that details what action must have been executed for this perceptual schema to be invoked. Essentially, these describe how a POMDP observation over perceptual propositions is generated after we execute an action and arrive at a successor state.

Giving an example, the following schema expresses that *dora* cannot observe a `model` $?m$ at a `place` `?loc` unless she has a photo of $?m$ in a `model-slot`.

```
(:observe reset_model_observations__on_state
```

```
:parameters
(?loc - place ?m - model)

:execution
()

:precondition
(and (observed_model_at_place ?loc ?m)
     (forall (?s - model-slot)
             (not (= (foregrounded-model ?s) ?m))) )


:effect
(and (not (observed_model_at_place ?loc ?m)))
)
```

The next schema expresses that *dora* cannot have perceptions about the `models` that are present in a place unless she has just executed an `explore-place` action.

```
(:observe reset_model_observations__on_execution
          :parameters
          (?loc - place ?m - model)

          :execution
          (not (explore-place ?loc))

          :precondition
          (and (observed_model_at_place ?loc ?m))

          :effect
          (and (not (observed_model_at_place ?loc ?m)))
          )
```

Finally, when *dora* does execute an `explore-place` action, we give schemata that describe the likelihood of her observing a `feature` or `widget` at the `place` she explored. In the case of a `feature` we have.

```
(:observe model_feature
          :parameters
          (?location - place ?l - label ?model - feature)

          :execution
          (explore_place ?location)

          :precondition
          (and
               (exists (?s - slot) (= (foregrounded-model ?s) ?model))

               )
          :effect
          (and
           (when (and (featured-at ?model ?location)
                      (= (labelled ?location) ?l))
             (probabilistic
              (probability__observe_feature_at_place_with_label
               __if_true ?location ?model ?l)
              (observed_model_at_place ?location ?model)
              )
             )
```

```
        (when (and (not (featured-at ?model ?location))
                   (= (labelled ?location) ?l))
          (probabilistic
           (probability--observe_feature_at_place_with_label
            --if_feature_false ?location ?model ?l)
           (observed_model_at_place ?location ?model)
           )
          )
        )
        )
)
```

Here, we have that when *dora* has explored the place ?`location`, she can observe features at ?`location` according to the *...observe_feature_at_place...* entries. For example, if there is some `feature` ?`model` at the explored ?`location` with `label` ?l , then with probability:

`probability--observe_feature_at_place_with_label--if_true` ?`location` ?`model` ?l)

dora makes the observation that ?`model` is at `place` ?`location`.

Finishing our example, in the case of perception with regards to a `widget` we have the following schema.

```
(:observe model_widget
          :parameters
          (?location - place ?l - label ?model - widget)

          :execution
          (explore-place ?location)

          :precondition
          (and
           (exists (?s - slot) (= (foregrounded-model ?s) ?model))

           (= (labelled ?location) ?l)
           )

          :effect
          (and
           (when (widget-location ?model ?location)
             (probabilistic
              (probability--observe_widget_model_at_label
               --if_true ?l ?model)
              (observed_model_at_place ?location ?model)
              )
             )

           (when (not (widget-location ?model ?location) )
             (probabilistic
              (probability--observe_widget_model_at_label
               --if_false ?l ?model)
              (observed_model_at_place ?location ?model)
              )
             )
           )
          )
```

7.5 Representations for Continual Planning

POMPDs, as introduced and modelled in DTPDDL in the previous sections, model uncertainty and how to reduce it using probability distributions over real world states and action outcomes. Reasoning about POMDPS is computationally quite complex. Therefore, as a complementary representation, we also try to represent gaps in knowledge explicitly in a single planner state, rather than a probability distribution. This representation is used in the Continual Planner. It is based on the SAS$^+$ formalism [4]. Here, instead of propositions, we use multi-valued state variables (MVSVs) v, each with an associated domain $vdom(v)$ describing the set of possible values $x \in vdom(v)$ that v may assume.

A *state* is defined as a function s associating variables v with values from their domain $vdom(v)$. If, for a given set of variables $\mathcal{V}$, s is not defined for all $v \in \mathcal{V}$, then s is called a *partial state* over $\mathcal{V}$.

There are several motivations for our using an SAS$^+$-based representation:

1. In recent years, SAS$^+$ has been shown to enable powerful reasoning techniques in planning algorithms, which has lead to systems based on this representation now dominating the International Planning Competition. Using a similar representation, we can exploit this development.

2. One of the explanations for the success of SAS$^+$ is the fact that it directly models natural mutual-exclusivity invariants between the values of MVSVs. For example, modelling the position of an object o using an MVSV $pos(o)$ explicitly states that this object is at one and only one position in any given state. This is not true for representations based on propositional logic, like STRIPS or PDDL, where any number of propositions $(posoloc_1)$, $(posoloc_2)$, .., $(posoloc_n)$ could be true in the same state.

3. In the context of our robotic architecture, the functional state representation of SAS$^+$ is also closer to the feature/value model used by other subarchitectures, in particular the representation uses by the Belief Model, from which planning states are generated. Roughly, each feature f of a belief b in a belief model is mapped onto a state variable $f(b)$. For example, if the belief model describes that a room has been categorised as a kitchen by attributing the *areaclass : kitchen* to a belief b, this would correspond to an assignment *areaclass(b) = kitchen* in a planning state.

4. The main reason for using an SAS$^+$-based representation is that we can employ it to explicitly model knowledge and gaps in knowledge,

so that the planner can efficiently reason about them. The rest of this section is dedicated to this aspect of our representation.

For the modelling needs of CogX, we have defined a specific formal language based on SAS$^+$ the multiagent planning language MAPL [8]. In MAPL, as in other planning formalisms, the general rules of the world are specified in a *planning domain* whereas a specific problem to be planned for in that domain in specified by a *planning task*.

A planning domain is a tuple $\mathcal{D} = (\mathcal{A}, \mathcal{V}, \mathcal{C}, \mathcal{E})$ consisting of agents $\mathcal{A}$, state variables $\mathcal{V}$, constants $\mathcal{C}$, and events $\mathcal{E}$.

MAPL states s are allowed to be partially defined, i. e., some MVSV values may be "unknown".

If, for some variable v currently unknown, possible candidate values are known, those can nevertheless be reasoned about in the planning state. The domain $vdom(v)$ is explicitly represented and can be changed by planning operators. For example, after unsuccessfully searching for an object o in a room r, o may be removed from $vdom(pos(o))$. If $vdom(v)$ is a singleton set $\{x\}$, then v will be set to x.

To enable reasoning about knowledge gaps and their filling explicitly, we introduce specific so-called *Kval* variables Kval_v with $vdom(\mathsf{Kval}_v) = \top, \bot$. The correspondence between a variable v and its Kval variable Kval_v is defined as follows: If $s(v)$ is defined with some value x then $s(\mathsf{Kval}_v) = \top$. This, of course, implies that if $s(\mathsf{Kval}_v) = \bot$ then $s(v)$ must be undefined.

The converse, however, is not true: We may want to describe a future state in which the value of variable v will be known, i. e., in which $\mathsf{Kval}_v = \top$, without being able to name that value, yet. While it is the nature of knowledge gaps that an agent cannot know in advance how exactly it will be filled, it is nevertheless crucial that the agent can reason about how to bring this about. To this end, MAPL uses Kval variables as epistemic effects of sensing actions.

In MAPL, a *sensor model* is an action that has an epistemic effects on the agent executing it.

For example, the action of running a room categorisation algorithm in a room is modelled in MAPL as follows:

```
(:sensor categorise_room
 :agent (?a - agent)
 :parameters (?r - room ?loc - place)
 :precondition (and
    (= (pos ?a) ?loc)
    (contains ?r ?loc))
 :sense (areaclass ?r)
)
```

In words, this sensor model describes that an agent can sense the area class of a room, i.e. its being a kitchen, office or hallway, once the agent is at a place that belongs to the room in question. At planning time, the

outcome of observing *areaclass(r)* is yet unknown, therefore the effect of `categorise_room(r,loc)` is formally described as $\mathsf{Kval}_{areaclass(r)} = \top$.

We distinguish two possible kinds of epistemic effects of a sensor model: (full) recognition and (binary) disambiguation. When, as in the above example, an agent is able to determine the value of a state variable v as soon as the preconditions of the sensor model `categorise_room` are satisfied and regardless of the value of v, we say that `categorise_room` fully recognises v.

Sometimes, however, sensing can only help to confirm or rule out a certain value $x \in vdom(v)$. This, we call binary disambiguation. Consider, e. g., a sensor model that models localisation of objects in rooms.

```
(:sensor localise_object
 :agent (?a - agent)
 :parameters (?o - object ?loc - place)
 :precondition
    (= (pos ?a) ?loc)
 :sense (= (pos ?o) ?l)
)
```

Binary sensor models do not guarantee that a value will be known after a sensing action. Yet, they guarantee that the domain of the corresponding MVSV will be smaller after the perception. Thus, binary sensor models can be used by the planner to generate exploratory behaviour in which the set of candidate values for a knowledge gaps is reduced repeatedly, until the true value is sensed or inferred.

Kval variables can appear in goal formulae as well, so that we can conveniently express *epistemic goals*, i.e. goals concerned with closing knowledge gap. Goal formulae can contain expressions in first-order logic, in particular conditionals and quantifiers. For example, the robot could have an epistemic goal to find out the categories for all rooms. This would be expressed by the following formula:

$\forall room.\mathsf{Kval}_{areaclass(room)} = \top$

A more complex epistemic goal like "explore all places belonging to rooms yet uncategorised" would be expressed as:

$\forall place.(\exists room.contains(place,\ room) \wedge \mathsf{Kval}_{areaclass(room)} = \bot) \rightarrow explored(place).$

MAPL is designed to be used by a *continual planner*, i. e., a planner that interleaves planning and execution deliberately and adapts its plans to a changing world state.

Interestingly, when planning for an epistemic goal that uses a quantified formula, goal the planner will also re-evaluate this goal when new instantations become possible. For example, for the last goal shown the planner will autonomously adapt its plan whenever new places and rooms are discovered.

A (slightly simplified) example of a plan using sensing actions that satisfy epistemic goals is given in Figure 11 on the following page.

In the George scenario and in our next instantiation of Dora, information will not only be obtained by sensing, but also through interaction with

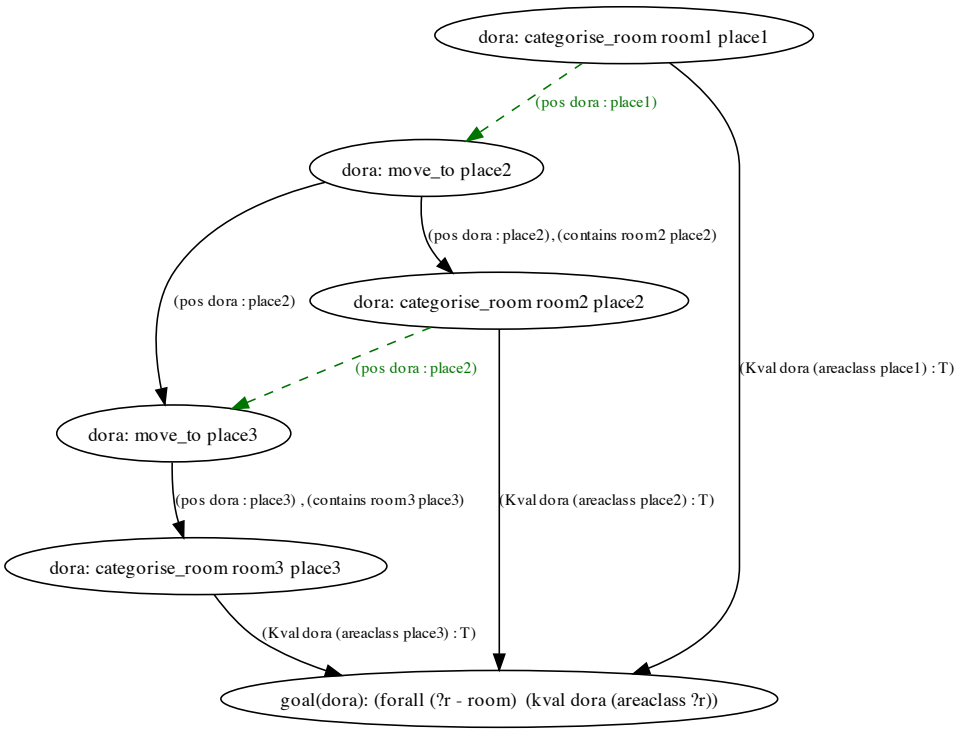


Figure 11: A plan using sensory actions to satisfy epistemic goals

humans. To plan for such multiagent interactions the robot must also reason about the knowledge of the other agents. We can express nested beliefs using MVSVs as well, e.g., "the robot R believes that human H believes that object o is a pen" is modelled as $\mathsf{K}[/R:, H]type(o) = pen$.

Knowledge gaps may arise in several variants when nested beliefs are used, depending on which agent is ignorant of the other's belief. Again, with MVSVs we can represent the differences succinctly using agent-specific "unknown" symbols. Consider, e.g., the difference between the statements "R knows that H does not know the location of the cornflakes" ($\mathit{Kval}^{R,H}_{pos(cornflakes)} = \bot^H$) and "R does not know if H knows the location of the cornflakes" ($(\mathit{Kval}^{R,H}_{pos(cornflakes)} = \bot^H)$.

Note that in contrast to classical epistemic logics, the MAPL representation explicitly represents the facts that both statements are mutually exclusive.

Just like sensing actions are modelled using standard *Kval* variables, we can use nested *Kval* variables to describe *speech acts*. In particular, we can describe *wh-questions* and answers to them ("where", "what colour", etc.) by modelling the appropriate nested belief effects. (Note: the planner was not used for dialogue planning in the George system as presented in this

paper, but will be in its next instantiation).

7.6 Summary: Representing and reasoning about knowledge gaps in planning

The representations introduced in this chapter relate to several aspects of the "typology of incompleteness" discussed in Section 2.

When possible values for an MVSV are known or a probabilistic belief state provides several alternatives propositions that may be true in the world, this is a case of **state variable value uncertainty**. In a POMDP, this uncertainty will be represented quantitatively, in the SAS$^+$-based representation used by the continual planner it is qualitative.

If no information is available about possible values of the unknown variable yet, we have a case of **state novelty**. This knowledge gap can be made explicit in MAPL, by setting the corresponding Kval variable to $\bot$, i. e., unknown.

We can also use the categories introduced in Section 2 to characterise the two different sensor models of MAPL: Binary sensor models have **effect value uncertainty**. It confirms or rules out a specific value from the known set of possible values of a state variable v. Sensor models for recognition do not need to refer to $vdom(v)$ because they abstract from the possible specific outcomes completely, only asserting $\mathit{Kval}_v = \top$. Therefore they also provide a way to deal with **effect novelty**.

The axis **multi-agent epistemic status** of the typology is covered by MAPL's capability to distinguish knowledge and knowledge gaps according to the agent they refer to. As exemplified in Section 7.5, we can distinguish whether an information is unknown to the robot or known to be unknown to another agent. Epistemic goals can refer to yet unattained shared beliefs and formal representations of speech acts model how such shared beliefs can be reached.

8 Representations of Cross-Modal Beliefs

Cross-modal beliefs rely on the particular representations used for learning in a cross-modal setting. These representations along with the cross-modal learning enable the robot to, based on interaction with the environment and people, extends its current knowledge by learning about the relationships between symbols and features that arise from the interpretation of different modalities. This involves processing of information from multiple modalities, which have to be adequately represented. One modality may exploit information from another to update its current representations, or several modalities together may be used to form representations of a certain concept. We focus here on the representations for encoding visual properties

and concepts that allow cross-modal learning through a dialogue with a human.

8.1 Representations for visual concepts

To efficiently store and generalize the observed information, the visual concepts are represented as generative models. These generative models take the form of probability density functions (pdf) over the feature space, and are constructed in online fashion from new observations. In particular, we apply the online Kernel Density Estimator (oKDE) [22] to construct these models. The oKDE estimates the probability density functions by a mixture of Gaussians, is able to adapt using only a single data-point at a time, automatically adjusts its complexity and does not assume specific requirements on the target distribution. A particularly important feature of the oKDE is that is allows adaptation from the positive as well as negative examples [23]. Specifically, the oKDE is defined by the so-called model of the observed samples $\mathbf{S}_{\text{model}}$,

$$\mathbf{S}_{\text{model}} = \{p_s(\mathbf{x}), \{q_i(\mathbf{x})\}_{i=1:N}\}, \tag{50}$$

which is composed of the *sample distribution*, a compressed model of the observed samples, $p_s(\mathbf{x})$ and of a *detailed model* $q_i(\mathbf{x})$, a necessary information for efficient adaptation of the compressed model[12]. The *sample distribution* is modelled by a mixture of Gaussians

$$p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x} - \mathbf{x}_i), \tag{51}$$

where

$$\phi_{\mathbf{\Sigma}}(\mathbf{x} - \mu) = (2\pi)^{-\frac{d}{2}} |\mathbf{\Sigma}|^{-\frac{1}{2}} e^{(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu))} \tag{52}$$

is a Gaussian kernel centered at μ with covariance matrix $\mathbf{\Sigma}$. The kernel density estimate (KDE) is then defined as a convolution of $p_s(\mathbf{x})$ by a kernel with a covariance matrix (bandwidth) $\mathbf{H}$ (see Figure 12):

$$p_{\text{KDE}}(\mathbf{x}) = \phi_{\mathbf{H}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}+\mathbf{\Sigma}_{si}}(\mathbf{x} - \mathbf{x}_i). \tag{53}$$

Note that the online KDE from (53) is the representation by which we operate when encoding and calculating beliefs of the observed data. Figure 13 demonstrates the power of the oKDE in estimating complex distributions from sequences of data.

[12] we will not go into details about the detailed model here, since it exceeds the scope of this document and we refer to [22] for more details.

Figure 12: Calculation of the KDE $p_{\text{KDE}}(\mathbf{x})$ through a convolution of the sample distribution $p_s(\mathbf{x})$ with a kernel. The upward arrows depict components in the sample distribution with zero covariance – Dirac-delta functions.

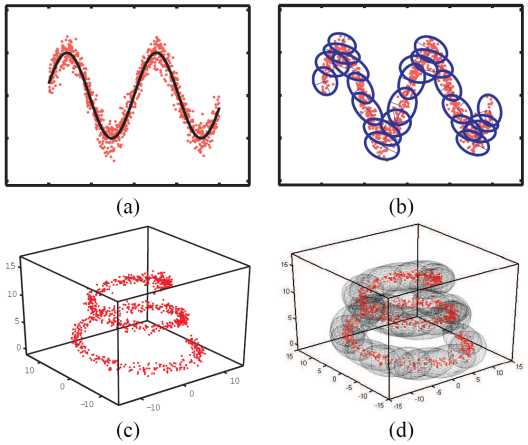
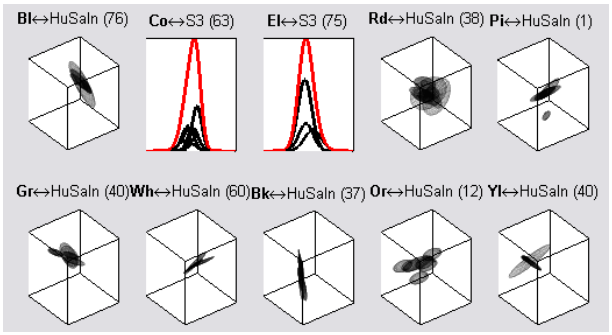


Figure 13: First row shows the sinusoidal distribution and the second row shows the spiral distribution. Left column shows the reference distributions and the right column shows the estimated distribution using oKDE after observing a 1000 samples.

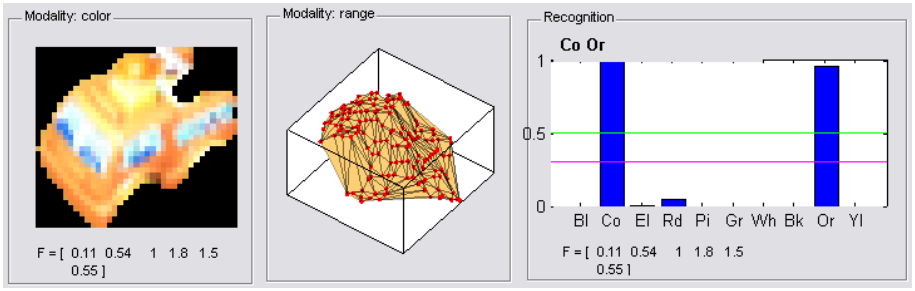
The continuous learning proceeds by extracting the visual data in a form of a highdimensional features (e.g., multiple 1D features relating to shape, texture, color and intensity of the observed object) and oKDE is used to estimate the pdf in this high-dimensional feature space. In this respect, the distributions model the knowledge incompleteness in terms of the **state value uncertainty** as described in Section 2. However, concepts such as *color red* reside only within lower dimensional subspace spanned only by features that relate to color (and not texture or shape). Therefore, during online learning, this unknown subspace constitutes a **structural novelty** and has to be identified to provide best performance. This is achieved by determining for a set of mutually exclusive concepts (e.g., colors green, blue, orange, etc.). We assume that this corresponds to the subspace which minimizes the overlap of the corresponding distributions. The overlap between the distributions is measured using the multivariate Hellinger distance [22].

Therefore, during online operation, a multivariate generative model is continually maintained for each of the visual concepts and for mutually exclusive sets of concepts the feature subspace is continually being determined.

The set of mutually exclusive concepts can then be used to construct a Bayesian classifier in the recognition phase, when the robot is generating a description of a particular object in terms of its color, shape, etc. An example of the learnt models from a real-life experiment are shown in Figure 14a and an example of classification of an observed object by the constructed Bayes classifier is shown in Figure 14b.



(a)



(b)

Figure 14: (a) Example of the models estimated using the oKDE and the feature selection algorithm. Note that some concepts are modelled by 3D distributions (e.g., "blue" which is denoted by "Bl"), while others (e.g., compact which is denoted by "Co") is modelled by 1D distributions. (b) From left to right: example of a segmented toy car, the extracted range data, and the results from the Bayes classifier constructed from the models in (a). The object is classified as "compact" and "orange".

Note that, since the system is operating in an online manner, the closed-world assumption can not be assumed. At every step of learning, the system should also take into the account that there might exist a yet "unknown model", which might better explain the robot's observation – the system should thus be aware of the **uncertainty about state model complexity**. In the following we describe how the the "unknown model" is probabilistically integrated into the system.

8.1.1 Accounting for unknown model

While maintaining good models of the visual concepts and being able to adapt those models is crucial for the robots online operation, the ability to detect gaps in the knowledge presented by these models is equally important. Generally speaking the robot collects the visual information about its environment as follows. First it determines a region in an image which contains the interesting information, then it "segments" that region and extracts the feature values z from which it later builds models of objects, concepts, etc. The visual information may be ambiguous by itself, and segmentation may not always be successful. We will assume that some measure of how well the segmentation was carried out exists and we will denote it by $s \in [0, 1]$. High values of s (around one) mean high confidence that a good observation z was obtained, while low values relate to low confidence.

Let $m \in \{m_k, m_u\}$ denote two possible events: (i) the observation came from a existing internal model m_k, and (ii) the observation came from an unknown model m_u. We define the knowledge model as a probability of observation z, given the confidence score s:

$$p(z|s) = p(z|m_k, s)p(m_k|s) + p(z|m_u, s)p(m_u|s). \tag{54}$$

The function $p(z|m_k, s)$ is the probability of explaining z given that z comes from one of the learnt models, $p(m_k|s)$ is the a priori probability of any learnt model given the observer's score s. The function $p(z|m_u, s)$ is the probability of z corresponding to the unknown model, and $p(m_u|s)$ is the probability of the model "unknown" given the score s.

Assume that the robot has learnt K separate alternative internal models $M = \{M_i\}_{i=1:K}$ from previous observations. The probability $p(z|m_k, s)$ can then be further decomposed in terms of these K models,

$$p(z|m_k, s) = \sum_{i=1}^{K} p(z|M_i, m_k, s)p(M_i|m_k, s). \tag{55}$$

If we define the "unknown" model by M_0 and set $p(z|m_u, s) = p(z|M_0, m_u, s)p(M_0|m_u, s)$, then (54) becomes

$$\begin{aligned} p(z|s) = p(m_k|s) \sum_{i=1}^{K} p(z|M_i, m_k, s)p(M_i|m_k, s) \\ +p(m_u|s)p(z|M_0, m_u, s)p(M_0|m_u, s). \end{aligned} \tag{56}$$

Note that the "unknown model", M_0, accounts for a poor classification, by which we mean that none of the learnt models supports the observation z strongly enough. We assume that the probability of this event is uniformly distributed over the feature space, which means that we can define the likelihood of model M_0, given observation z by a uniform distribution, i.e.,

$p(z|M_0, m_u, s) = \mathcal{U}(z)$. Note also, that the only possible unknown model comes from the class M_0, therefore $p(M_0|m_u, s) = 1$.

The observation z can be classified into the class M_i which maximizes the a posteriori probability (AP). The a posteriori probability of a class M_i is calculated as

$$p(M_i|z, s) \propto p(z|M_i, m, s)p(M_i|m, s)p(m|s), \tag{57}$$

where $m = m_k$ for $i \in [1, K]$ and $m = m_u$ for $i = 0$.

The gap in knowledge can be discovered through inspection of the AP distribution, which effectively reflects the **uncertainty about the state model complexity**. In particular, if the AP distribution exhibits an *ambiguous classification* of the observation z, or classifies it as an "unknown" (or unaccounted), then this is a good indication that we are dealing with a gap in knowledge.

In our implementations, the distribution of each i-th alternative of the known model $p(z|M_i, m_k, s)$ is continually updated by the oKDE [22], while the a priori probability $p(M_i|m_k, s)$ for each model is calculated from the frequency at which each of the alternative classes M_i, $i > 0$, has been observed. The a priori probability of an unknown model (and implicitly of a known model), $p(m_{\mathrm{u}}|s)$ is assumed non-stationary in that it changes with time. The following function decreases the "unknown" class probability with increasing number of observations N and increases this probability if the observer's certainty score s is low[13]:

$$p(m_{\mathrm{u}}|s) \quad = \quad e^{-0.5(\frac{N}{\sigma_N})^2}. \tag{58}$$

With above definitions, the knowledge model is completely defined and allows discovery of knowledge gaps.

8.1.2 Example of a probabilistic knowledge model

For a better visualization of the knowledge update and gap discovery we will restrict our example to a one-dimensional case. We will also use this example to better relate the types of knowledge incompleteness to the definitions from Section 2. Fig. 15 illustrates detection and filling of knowledge gaps for three cases (feature values) denoted by the circle, the diamond, and the square. The plots in the left column depict the models, the posteriori pdfs, and the recognition at a particular step in the learning process. The right column depicts the situation after the system has updated these models considering the detected knowledge gaps and the answers from the tutor. Note that the pdfs over the feature values account for the **state value uncertainty** in

[13]For example, in visual learning, the observer's certainty score s might reflect the quality of the visual data from which the visual features are extracted.

the models, while the a posteriori pdfs account for the **uncertainty about the state model complexity**.

The circle represents a yellow object. Since the yellow colour has not been presented to the robot before, the corresponding model has not yet been learned and the feature value fails in a not yet modelled area, therefore this value is best explained by the "unknown model", which has the highest a posteriori probability. The robot asks the tutor "*What colour is this object?*", and after the tutor provides the correct information, the robot initializes a model for yellow colour. We say that a **state novelty** has been detected by the robot, and after the human's explanation, the **structural novelty** has also been registered. Note, however, that since only a single sample does not suffice to build a reliable representation, the yellow colour will only be able to be recognized after some additional yellow object is observed.

The feature value denoted by a diamond is best explained by a green model, however this recognition is not very reliable, therefore the robot asks the tutor: "*Is this object green?*" to verify its belief. This question is triggered by a high **state value uncertainty**. After the tutor replies "*No. It is blue.*", the robot first unlearns the representation of green and updates the representation of blue. The corrected representations, depicted in the pdfs in the right column , then enable the correct recognition as indicated by the second bar plot in the right column of the Fig. 15.

The last case denoted by the square shows another example of non-reliable recognition, which triggers the additional clarification question to the tutor: "*Is this object blue?*" After the robot gets a positive answer, it updates the representation of blue, which increases the probability of the recognition. In this case, the question is triggered by a high **state value uncertainty**, which borders on possibility of a **state novelty**. However, after the tutor's answer, the **state value uncertainty** is decreased without creating a novel state as in the case of the feature value denoted by a diamond.

9 Conclusion

In this report we have given a detailed formal account of the representations we employ in different modalities, including models of spatial knowledge, dialogue state, multi-agent beliefs, action effects, visual information, and the relationships between features in different modalities. In each case we have given the basic representational forms, and then described how these naturally incorporate uncertainty and/or gaps, or how they be extended to do so. In particular we have developed a way of posing the binding problem as one of probabilistic inference. This allows us to represent quantitative uncertainty in the way information may be related between modalities. We

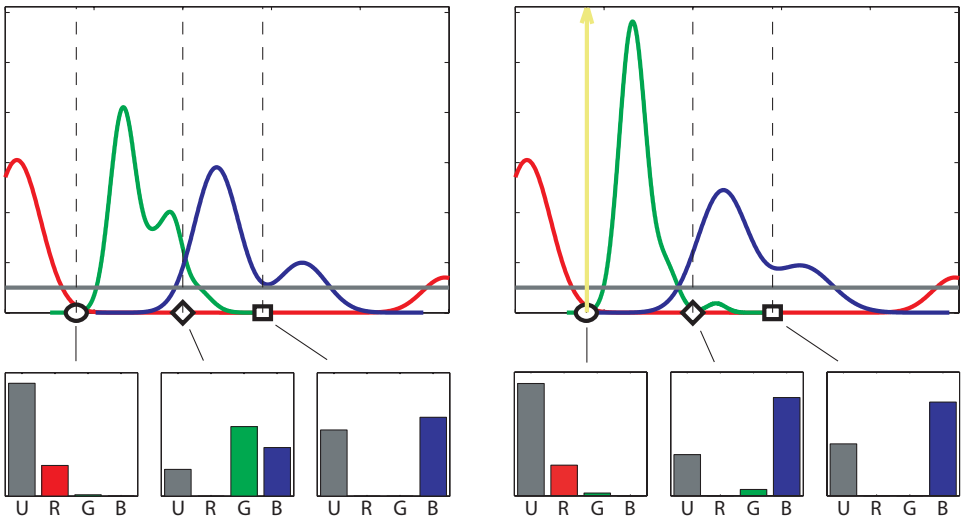


Figure 15: Example of detecting the knowledge gaps and updating the 1D KDE representations. Top row: probability distributions for three colours (red, green, blue lines) and unknown model (gray line) in 1D feature space. Bottom row: a posteriori probabilities for the unknown model (U) and three colours (R, G, B) for three feature values denoted by the circle, the diamond and the square. Left column: before updates, right column: after updates.

refer to these representations as multi-modal beliefs, but they are essentially amodal, logical representations of the environment, suitable for use in creating state descriptions needed by high level planning.

There are two common themes running through this story. We use qualitative representations of epistemic state in the multi-modal representations, in models of action effects in continual planning and in dialogue. Thus in these representations we can represent explicitly certain kinds of gaps, and how they change under action. In other modalities (vision, cross-modal representations, DT planning) we have employed a formalism that is essentially probabilistic, and which allows us to integrate evidence gradually from different modalities to reduce uncertainty. The representations in cross-modal learning are able to capture some open worldness, such as the presence of sensory data unmodelled by existing classes.

Finally we have also shown how these different types of uncertainty and gaps are broadly related in a typology. This typology is useful, because aside from anything else it assists in enabling us to spot gaps in our thinking about gaps. It is quite clear to us that several areas require considerably more work. Our approach to planning, for example, circumvents rather than addresses head on the issue of open-worldness. It may transpire that this is the best we can do, but we will work in the future on finding representations of action effects that model the effects of learning, which are inherently open-ended.

We are currently working on probabilistic models of manipulative effects (to be detailed in DR2.2), and on the use of POMDP models of dialogue management. So this document will be updated significantly by the end of the project. It is in this sense a living document that provides a basis for internal discussion on our representations, as well as a snapshot of where we are now in terms of modelling beliefs, gaps and uncertainty.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

References

[1] J. Allwood. An activity based approach to pragmatics. In H. Bunt and B. Black, editors, *Abduction, Belief and Context in Dialogue: Studies in Computational Pragmatics*, pages 47–80. John Benjamins, Amsterdam, The Netherlands, 2000.

[2] C. Areces. *Logic Engineering. The Case of Description and Hybrid Logics*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands, October 2000.

[3] A. Aydemir, A. Bishop, and P. Jensfelt. Simultaneous object class and pose estimation for mobile robotic applications with minimalistic recognition. In *Proc. of the International Conference on Robotics and Automation (ICRA'09)*, 2010.

[4] Christer Bäckström and Bernhard Nebel. Complexity results for SAS^+ planning. *Computiational Intelligence*, 11(4):625–655, 1995.

[5] J. Baldridge and G.-J. M. Kruijff. Coupling CCG and hybrid logic dependency semantics. In *ACL'02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 319–326, Philadelphia, PA, 2002. Association for Computational Linguistics.

[6] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I and II*. Athena Scientific, 2007.

[7] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–625, 2000.

[8] Michael Brenner and Bernhard Nebel. Continual planning and acting in dynamic multiagent environments. *Journal of Autonomous Agents and Multiagent Systems*, 19(3):297–331, 2009.

[9] J. Carroll and S. Oepen. High efficiency realization for a wide-coverage unification grammar. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP'05)*, pages 165–176, 2005.

[10] H. Clark. *Using Language*. Cambridge University Press, Cambridge, UK, 1996.

[11] H. H. Clark and E. F. Schaefer. Contributing to discourse. *Cognitive Science*, 13:259–294, 1989.

[12] N. Cristianini and J. S. Taylor. *An Introduction to SVMs and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[13] R. Davis, H. Shrobe, and P. Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[14] Stefan Edelkamp and Jörg Hoffmann. PDDL2.2: The language for the classical part of the 4th international planning competition. Technical Report 195, Albert-Ludwigs-Universität, Institut für Informatik, Freiburg, Germany, 2004.

[15] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. The MIT Press, Cambridge, MA, USA, 1995.

[16] R.E. Fikes and N.J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. Technical Report 43r, AI Center, SRI International, 333 Ravenswood Ave, Menlo Park, CA 94025, May 1971. SRI Project 8259.

[17] John Folkesson, Patric Jensfelt, and Henrik Christensen. The m-space feature representation for slam. *IEEE Transactions on Robotics*, 23(5):1024–1035, October 2007.

[18] M. Fox and D. Long. PDDL 2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[19] H. P. Grice. Logic and conversation. In P. Cole and J. L. Morgan, editors, *Speech Acts: Syntax and Semantics, Volume 3*, pages 41–58. Academic Press, New York, 1975.

[20] Nick Hawes, Hendrik Zender, Kristoffer Sjöö, Michael Brenner, Geert-Jan M. Kruijff, and Patric Jensfelt. Planning and acting with an integrated sense of space. In Alexander Ferrein, Josef Pauli, Nils T. Siebel, and Gerald Steinbauer, editors, *HYCAS 2009: 1st International Workshop on Hybrid Control of Autonomous Systems – Integrating Learning,*

Deliberation and Reactive Control, pages 25–32, Pasadena, CA, USA, July 2009.

[21] R. A. Howard. *Dynamic Probabilistic Systems*. John Wiley & Sons, New York., 1971.

[22] M. Kristan and A. Leonardis. Multivariate online kernel density estimation. In *Computer Vision Winter Workshop*, page accepted, 2010.

[23] M. Kristan, D. Skočaj, and A. Leonardis. Online kernel density estimation for interactive learning. *Image and Vision Computing*, 2009.

[24] G.-J. M. Kruijff. *A Categorial-Modal Logical Architecture of Informativity: Dependency Grammar Logic & Information Structure*. PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, April 2001.

[25] G.J.M. Kruijff, M. Brenner, and N.A. Hawes. Continual planning for cross-modal situated clarification in human-robot interaction. In *Proceedings of the 17th International Symposium on Robot and Human Interactive Communication (RO-MAN 2008)*, Munich, Germany, 2008.

[26] P. Lison and G.-J. M. Kruijff. Salience-driven contextual priming of speech recognition for human-robot interaction. In *Proceedings of ECAI 2008*, Athens, Greece, 2008.

[27] P. Lison and G.J.M. Kruijff. Efficient parsing of spoken inputs for human-robot interaction. In *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 09)*, Toyama, Japan, 2009.

[28] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[29] D. McDermott. PDDL – the planning domain definition language. Technical Report TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[30] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[31] S. Oepen and J. Carroll. Ambiguity packing in constraint-based parsing: Practical results. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP 2000)*, pages 162–169, 2000.

[32] A. Pronobis and B. Caputo. Confidence-based cue integration for visual place recognition. In *Proceedings of the IEEE/RSJ International*

Conference on Intelligent Robots and Systems (IROS07), San Diego, CA, USA, October 2007.

[33] A. Pronobis, O. Martinez Mozos, and B. Caputo. SVM-based discriminative accumulation scheme for place recognition. In *Proc. of ICRA'08*, Pasadena, CA, USA, 2008.

[34] Andrzej Pronobis, Kristoffer Sjöö, Alper Aydemir, Adrian N. Bishop, and Patric Jensfelt. Representing spatial knowledge in mobile cognitive systems. Technical Report TRITA-CSC-CV 2010:1 CVAP 316, Kungliga Tekniska Högskolan, CVAP/CAS, March 2010.

[35] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *4th International Symposium on Robotics Research*, 1987.

[36] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, Massachusetts 02142, 2005. ISBN 0-262-20162-3.

[37] H. L. S. Younes. Extending pddl to model stochastic decision processes. In *ICAPS, Workshop on PDDL*, pages 95–103, 2003.

[38] H. L. S. Younes and M. Littman. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2004.

[39] Hendrik Zender, Óscar Martínez Mozos, Patric Jensfelt, Geert-Jan M. Kruijff, and Wolfram Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, June 2008.

A Grammar for CogX Decision-Theoretic Planning Domain Definition Language (DTPDDL0.9β)

The 3rd Planning Domain Definition Language (PDDL3.0 – pronounced "pea-diddle") is the language of choice of the symbolic planning community. This language and its predecessors have been developed for and adopted at the International Planning Competitions since 1998. Moreover, a PDDL-based language formed the basis of domain and problem description in the planning subarchitecture of CoSy.

This section gives the Extended Backus Normal Form (EBNF) grammar for defining decision-theoretic domains and problems called DTPDDL0.9β – pronounced "deeteepee-diddle". In particular, we extend PPDDL1.0 pronounced "pea-two-diddle" – a language for describing probabilistic planning problems that has been used since 2004 in International Planning Competitions [38] – to contain syntactic elements for describing domains and corresponding problems that feature partial observability. Our work draws heavily on the work of [37] that sought extensions of PDDL for modelling stochastic decision processes.

A.1 Domain Definition

```
<domain>              ::= (define (domain <name>)
                                [<require-def>]
                                [<types-def>]:typing
                                [<constants-def>]
                                [<s-functions-def>]:fluents
                                [<o-functions-def>]:fluents
                                [<predicates-def>]
                                [<observations-def>]
                                <structure-def>*)
<require-def>         ::= (:requirements <require-key>+)
<require-key>         ::= :strips
<require-key>         ::= :fluents
<require-key>         ::= :typing
<require-key>         ::= :equality
<require-key>         ::= :existential-preconditions
<require-key>         ::= :universal-preconditions
<require-key>         ::= :quantified-preconditions
  Sugar for including :existential-preconditions and :universal-preconditions
<require-key>         ::= :universal-effects
<require-key>         ::= :conditional-effects
<require-key>         ::= :probabilistic-effects
<require-key>         ::= :partial-observability
<require-key>         ::= :universal-unwinding
<s-functions-def>     ::=:fluents
  (:s-functions <function typed list (atomic s-function skeleton)>)
<o-functions-def>     ::=:fluents
  (:o-functions <function typed list (atomic o-function skeleton)>)
<atomic s-function skeleton>
                      ::= (<s-function-symbol> <typed list (variable)>)
<atomic o-function skeleton>
                      ::= (<o-function-symbol> <typed list (variable)>)
<function typed list (x)>
                      ::= x*
```

```
<function typed list (x)>
                      ::= :typing x+- <function-type> <function typed list (x)>
<typed list (x)>      ::= x*
<typed list (x)>      ::= :typing x+- <type> <typed list (x)>
<function-type>       ::= int|float|double
<emptyOr (x)>         ::= ()
<emptyOr (x)>         ::= x
<type>                ::= (either <primitive-type>+)
<type>                ::= <primitive-type>
<types-def>           ::= (:types <typed list (name)>)
<constants-def>       ::= (:constants <typed list (name)>)
<predicates-def>      ::= (:predicates <atomic s-formula skeleton>+)
<observations-def>    ::= (:percepts <atomic o-formula skeleton>+)
<atomic s-formula skeleton>
                      ::= (<predicate> <typed list (variable)>)
<atomic o-formula skeleton>
                      ::= (<observation> <typed list (variable)>)
<predicate>           ::= <name>
<observation>         ::= <name>
<o-function-symbol>   ::= <name>
<s-function-symbol>   ::= <name>
<variable>            ::= ?<name>
<structure-def>       ::= <action-def>
<structure-def>       ::= :partial-observability <observation-def>
```

A.1.1 Actions

```
<action-def>              ::= (:action <action-symbol>
                                   :parameters (<typed list (variable)>)
                                   <action-def body>)
<action-def body>         ::= [:precondition <emptyOr (pre-GD)>]
                              [:effect <emptyOr (s-effect)>]
<action-symbol>           ::= <name>
<pre-GD>                  ::= (and <pre-GD>*)
<pre-GD>                  ::= :universal-preconditions
  (forall (<typed list (var)>) <pre-GD>)
<pre-GD>                  ::= :existential-preconditions
   (exists (<typed list (var)>) <pre-GD>)
<pre-GD>                  ::= <GD>
<GD>                      ::= <atomic s-formula (term)>
<GD>                      ::= (and <GD>*)
<GD>                      ::= :fluents <s-f-comp>
<atomic s-formula(t)>     ::= (<predicate> t*)
<term>                    ::= <name>
<term>                    ::= <variable>
<s-f-comp>                ::= (<binary-comp> <s-f-exp> <s-f-exp>)
<s-f-exp>                 ::= <number>
<s-f-exp>                 ::= (<binary-op> <s-f-exp> <s-f-exp>)
<s-f-exp>                 ::= (<multi-op> <s-f-exp> <s-f-exp>+)
<s-f-exp>                 ::= (- <s-f-exp>)
<s-f-exp>                 ::= <s-f-head>
<s-f-head>                ::= (<s-function-symbol> <term>*)
<s-f-head>                ::= <s-function-symbol>
<binary-op>               ::= <multi-op>
<binary-op>               ::= -
<binary-op>               ::= /
<multi-op>                ::= *
<multi-op>                ::= +
<binary-comp>             ::= >
<binary-comp>             ::= <
```

```
<binary-comp>             ::= =
<binary-comp>             ::= >=
<binary-comp>             ::= <=
<number>                  ::= int|float|double
<prob>                    ::= float|double (≥ 0, ≤ 1)
<s-effect>                ::= (and <c-s-effect>*)
<s-effect>                ::= <c-s-effect>
<s-effect>                ::= <q-s-effect>
<c-s-effect>              ::= :conditional−effects (when <GD> <s-effect>)
<c-s-effect>              ::= :universal−effects (forall (<typed list (var)>) <s-effect>)
<c-s-effect>              ::= :probabilistic−effects (probabilistic <prob> <s-effect>)
<c-s-effect>              ::= :probabilistic−effects:universal−unwinding
  (probabilistic (for-each (<typed list (x)>) <s-f-head> <s-effect> ))
<c-s-effect>              ::= <p-s-effect>
<p-s-effect>              ::= <atomic s-formula(term)>
<p-s-effect>              ::= (not <atomic s-formula(term)>)
<p-s-effect>              ::= :fluents(<assign-op> <s-f-head> <s-f-exp>)
<assign-op>               ::= assign
<assign-op>               ::= scale-up
<assign-op>               ::= scale-down
<assign-op>               ::= increase
<assign-op>               ::= decrease
```

A.1.2 Observations

In classical and probabilistic planning all predicates are fully observable. In CogX we are concerned with decision-theoretic planning domains where the truth value of perceptual propositions are all the agent has in order to determine its beliefs about the world. Here we give the grammar for observation schema that determine the truth values of perceptual propositions.

```
<observation-def>         ::= (:observe <o-symbol>
                                  :parameters (<typed list (variable)>)
                                  <o-def body>)
<o-def body>              ::= [:state <emptyOr (pre-GD)>]
                              [:action <atomic action(term)> ]
                              [:effect <emptyOr (o-effect)>]
<o-symbol>                ::= <name>
<atomic action(t)>        ::= (<action-symbol> t*)
<o-effect>                ::= (and <c-o-effect>*)
<o-effect>                ::= <c-o-effect>
<c-o-effect>              ::= :conditional−effects (when <GD> <o-effect>)
<c-o-effect>              ::= :universal−effects (forall (<typed list (var)>) <o-effect>)
<c-o-effect>              ::= :probabilistic−effects (probabilistic <prob> <o-effect>)
<c-o-effect>              ::= :probabilistic−effects:universal−unwinding
  (probabilistic (for-each (<typed list (x)>) <s-f-head> <o-effect> ))
<c-o-effect>              ::= <p-o-effect>
<atomic o-formula(t)>     ::= (<observation> t*)
<p-o-effect>              ::= <atomic o-formula(term)>
<p-o-effect>              ::= (not <atomic o-formula(term)>)
<p-o-effect>              ::= :fluents(<assign-op> <o-f-head> <o-f-exp>)
<o-f-comp>                ::= (<binary-comp> <o-f-exp> <o-f-exp>)
<o-f-exp>                 ::= <number>
<o-f-exp>                 ::= (<binary-op> <o-f-exp> <o-f-exp>)
<o-f-exp>                 ::= (<multi-op> <o-f-exp> <o-f-exp>+)
<o-f-exp>                 ::= (- <o-f-exp>)
<o-f-exp>                 ::= <o-f-head>
<o-f-head>                ::= (<o-function-symbol> <term>*)
```

```
<o-f-head>              ::= <o-function-symbol>
```

A.2 Problem Definition

```
<problem>               ::= (define (problem <name>)
                                (:domain <name>)
                                [ <object declaration> ]
                                <init>
                                <goal>
                                [ <metric-spec> ] )
<object declaration>    ::= (:objects <typed list (name)>)
<init>                  ::= (:init <init-el>*)
<init-el>               ::= <literal (name)>
<init-el>               ::= :probabilistic-effects (probabilistic <prob> <init-el>*)
<init-el>               ::= (= <s-f-head> <number>)
<goal>                  ::= (:goal <pre-GD>)
<literal (t)>           ::= <atomic s-formula (t)>
<literal (t)>           ::= (not <atomic s-formula (t)>)
<metric-spec>           ::= (:metric <optimization> <metric-f-exp>)
<optimization>          ::= minimize
<optimization>          ::= maximize
<metric-f-exp>          ::= (<binary-op> <metric-f-exp> <metric-f-exp>)
<metric-f-exp>          ::= (<multi-op> <metric-f-exp> <metric-f-exp>+)
<metric-f-exp>          ::= (- <metric-f-exp>)
<metric-f-exp>          ::= <number>
<metric-f-exp>          ::= (<s-function-symbol> <name>*)
<metric-f-exp>          ::= <s-function-symbol>
```

A.2.1 Example from IPC-5 Tireworld

Here we demonstrate our POMDP domain definition language by giving an example of a tireworld problem from IPC-5 that has some partial observability.

```
;;;  Original Authors: Michael Littman and David Weissman  ;;;
;;;  Modified: Blai Bonet for IPC 2006 ;;;
;;;  Modified: Charles Gretton for CogX 2009 ;;;

(define (domain tire)
  (:requirements
      :partial-observability ;; Not in IPC-5 tireworld
      :fluents ;; Not in IPC-5  tireworld
      :universal-effects  ;; Not in IPC-5 tireworld
      :conditional-effects  ;; Not in IPC-5 tireworld

      :typing
      :strips
      :equality
      :probabilistic-effects)

  (:types location)

  (:predicates
   (vehicle-at ?loc - location)

   (spare-in ?loc - location)

   (road ?from - location ?to - location)
```

```
 (goal-location ?loc) ;; Not in IPC-5 tireworld

 (not-flattire)

 (hasspare)
 )

;; Note, here all the state-predicates are repeated except for
;; "(not-flattire)". A repeated state predicate in ":percepts" is
;; fully observable.
(:percepts

 ;; Do we know if we have a flat tire?
 (observe-not-flattire) ;; Not in IPC-5 tireworld

 ;; Fully observable -- i.e. follows state variable.
 (vehicle-at ?loc - location)

 ;; Fully observable -- i.e. follows state variable.
 (spare-in ?loc - location)

 ;; Fully observable -- i.e. follows state variable.
 (road ?from - location ?to - location)

 ;; Fully observable -- i.e. follows state variable.
 (goal-location ?loc)

 ;; Fully observable -- i.e. follows state variable.
 (hasspare)
 )

(:action move-car
  :parameters
  (?from - location ?to - location)

  :precondition
  (and
   (vehicle-at ?from)
   (road ?from ?to)
   (not-flattire))

  :effect
  (and
   (vehicle-at ?to)
   (not (vehicle-at ?from))
   (probabilistic 2/5 (not (not-flattire)))

   ;; Following was not in IPC-5 tireworld
   (forall
    (?loc - location)
    (when (and (goal-location ?loc)
               (= ?to ?loc))
          (increase (reward) 1000)))
   )
)
(:action loadtire
  :parameters (?loc - location)
  :precondition (and (vehicle-at ?loc)
                     (spare-in ?loc))
  :effect (and (hasspare) (not (spare-in ?loc)))
 )
(:action changetire
```

```
    :precondition (hasspare)
    :effect (probabilistic 1/2 (and (not (hasspare))
                                    (not-flattire)))
  )

  ;; Following perception was not in IPC-5 tireworld
  (:observe tire-status-after-move
   :parameters
   (?from - location ?to - location)

   :execution
   (move-car ?from ?to)

   :precondition
   ()

   :effect
   (and (when (not-flattire)
          (probabilistic 7/8 (observe-not-flattire)
                         1/8 (not (observe-not-flattire))))
        (when (not (not-flattire))
          (probabilistic 7/8 (not (observe-not-flattire))
                         1/8 (observe-not-flattire))))

   )
)

(define (problem tire_17_0_28460)
  (:domain tire)
  (:objects n0 n1 n2 n3 n4 n5 n6 n7 n8
            n9 n10 n11 n12 n13 n14 n15 n16 - location)
  (:init (vehicle-at n2)
         (road n0 n12) (road n12 n0)
         (road n0 n16) (road n16 n0)
         (road n1 n2) (road n2 n1)
         (road n1 n3) (road n3 n1)
         (road n3 n4) (road n4 n3)
         (road n3 n13) (road n13 n3)
         (road n3 n14) (road n14 n3)
         (road n5 n8) (road n8 n5)
         (road n5 n10) (road n10 n5)
         (road n5 n16) (road n16 n5)
         (road n6 n14) (road n14 n6)
         (road n7 n9) (road n9 n7)
         (road n7 n13) (road n13 n7)
         (road n8 n9) (road n9 n8)
         (road n9 n12) (road n12 n9)
         (road n9 n16) (road n16 n9)
         (road n10 n12) (road n12 n10)
         (road n10 n13) (road n13 n10)
         (road n11 n16) (road n16 n11)
         (road n12 n16) (road n16 n12)
         (road n13 n15) (road n15 n13)
         (road n14 n16) (road n16 n14)
         (spare-in n4)
         (spare-in n5)
         (spare-in n7)
         (spare-in n8)
         (spare-in n10)
         (spare-in n12)
         (spare-in n16)
         (probabilistic 8/9 (not-flattire))
```

```
        (goal-location n0)
  )

  (:metric maximize (reward))
)
```

B Annexes

B.1 Wyatt et al. "Self-Understanding and Self-Extension: A Systems and Representational Approach

Bibliography Jeremy L. Wyatt, Alper Aydemir, Michael Brenner, Marc Hanheide, Nick Hawes, Patric Jensfelt, Matej Kristan, Geert-Jan M. Kruijff, Pierre Lison, Andrzej Pronobis, Kristoffer Sjöö, Danijel Skočaj, Alen Vrečko, Hendrik Zender, Michael Zillich: Self-Understanding and Self-Extension: A Systems and Representational Approach, submitted to *IEEE Transactions on Autonomous Mental Development, Special Issue on Architectures and Representations.*

Abstract There are many different approaches to building a system that can engage in autonomous mental development. In this paper we present an approach based on what we term self- understanding, by which we mean the use of explicit representa- tion of and reasoning about what a system does and doesnt know, and how that understanding changes under action. We present a coherent architecture and a set of representations used in two robot systems that exhibit a limited degree of autonomous mental development, what we term self-extension. The contributions include: representations of gaps and uncertainty for specic kinds of knowledge, and a motivational and planning system for setting and achieving learning goals.

Relation to WP This paper describes the representations of beliefs about uncertainty and gaps in spatial information, planning, cross-modal information, and multi-modal information. It shows how these were used in the George and Dora systems during year 1. In particular it shows how to build a system that uses representations of beliefs several different modalities and how those change under action in a unified way.

B.2 Lison et al. "Belief Modelling for Situation Awareness in Human-Robot Interaction

Bibliography Pierre Lison, Carsten Ehrler and Geert-Jan M. Kruijff: *Belief Modelling for Situation Awareness in Human-Robot Interaction.* Extended report.

Abstract To interact naturally with humans, robots needs to be aware of their own surroundings. This awareness is usually encoded in some implicit or explicit representation of the situated context. In this research report, we present a new framework for constructing rich belief models of the robot's environment.

Key to our approach is the use of *Markov Logic* as a unified representation formalism. Markov Logic is a combination of first-order logic and probabilistic graphical models. Its expressive power allows us to capture both the rich relational structure of the environment and the uncertainty arising from the noise and incompleteness of low-level sensory data. Beliefs evolve dynamically over time, and are constructed by a three-fold iterative process of information fusion, refinement and abstraction. This process is reflected in distinct ontological categories. Links across these categories define the construction history by relating a belief to its ancestors. Beliefs are thus organised in a complex two-dimensional structure, with horizontal relations between belief dependents and vertical relations between belief relatives.

Beliefs also incorporate various contextual information such as spatio-temporal framing, multi-agent epistemic status, and saliency measures. Such rich annotation scheme allows us to easily interface beliefs with high-level cognitive functions such as action planning or communication. Beliefs can therefore be easily referenced, controlled and extended "top-down" by external processes to reach beyond the current perceptual horizon and include past, future or hypothetical knowledge.

Relation to WP This report describes the formal representations used to model multi-modal beliefs (and how they can be build). The approach extends the probabilistic binding approach developed for year 1.

A shorter version of the report has been submitted to RO-MAN 2010 (19th IEEE International Symposium on Robot and Human Interactive Communication), under the same title.

Self-Understanding & Self-Extension: A Systems and Representational Approach

Jeremy L. Wyatt, Alper Aydemir, Michael Brenner, Marc Hanheide, Nick Hawes, Patric Jensfelt, Matej Kristan, Geert-Jan M. Kruijff, Pierre Lison, Andrzej Pronobis, Kristoffer Sjöö, Danijel Skočaj, Alen Vrečko, Hendrik Zender, Michael Zillich

***Abstract*—There are many different approaches to building a system that can engage in autonomous mental development. In this paper we present an approach based on what we term *self-understanding*, by which we mean the use of explicit representation of and reasoning about what a system does and doesn't know, and how that understanding changes under action. We present a coherent architecture and a set of representations used in two robot systems that exhibit a limited degree of autonomous mental development, what we term *self-extension*. The contributions include: representations of gaps and uncertainty for specific kinds of knowledge, and a motivational and planning system for setting and achieving learning goals.**

***Index Terms*—robotics, robot learning, architectures, representations**

I. Introduction

WHAT is needed for an agent to learn in a truly autonomous fashion? One way is to give that agent knowledge of what it knows and doesn't know, and to make it reason with these representations to set its own *epistemic goals*. An epistemic goal is a goal to be in a certain knowledge state. In this paper we describe this representation and systems approach to autonomous mental development. We present an architecture, together with a set of representations that explicitly capture what the robot and other agents do and don't know at any one time, i.e. representations of their epistemic state. We also describe representations of how this epistemic state will change under action. Such representations with algorithms for reasoning about them we refer to as conferring a degree of *self-understanding*, and allow the construction of systems that are able to plan how to extend the knowledge they have of the environment, i.e. knowledge *self-extension*. We also describe a goal management system that allows the robot to choose quickly between different epistemic goals. We argue that such an approach will be necessary in the long term as robot systems become able to generate many goals for filling gaps in and reducing uncertainty in knowledge.

Jeremy L. Wyatt, Marc Hanheide and Nick Hawes are with the University of Birmingham, email: {jlw,nah,m.hanheide}@cs.bham.ac.uk

Michael Brenner is with Albert-Ludwigs-Universität, email: brenner@informatik.uni-freiburg.de

Pierre Lison, Geert-Jan M. Kruijff and Hendrik Zender are with DFKI, Saarbrücken, Germany, email: {plison,gj,zender}@dfki.de

Patric Jensfelt, Andrzej Pronobis, Kristoffer Sjöö and Alper Aydemir are with KTH Stockholm, email: {patric,krsj,pronobis,aydemir}@csc.kth.se

Matej Kristan, Alen Vrečko and Danijel Skočaj are with University of Ljubljana, email: {matej.kristan,alen.vrecko,danijel.skocaj}@fri.uni-lj.si

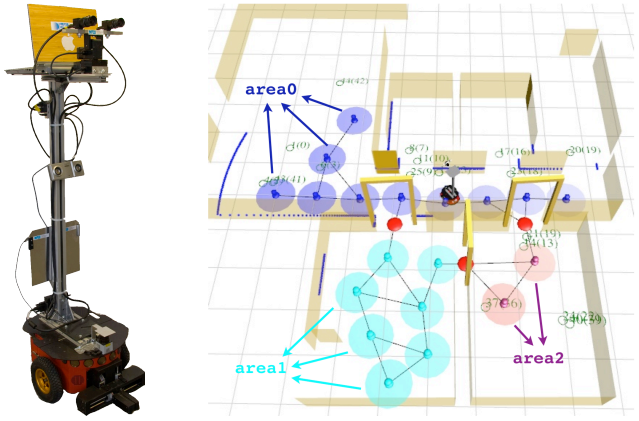
Michael Zillich is with Vienna University of Technology, email: zillich@acin.tuwien.ac.at

Manuscript received February 28, 2010

It is important to understand a little of the different types of incompleteness in knowledge. We use *incompleteness* as an umbrella term to cover many different types of *knowledge gaps* and *uncertainty about knowledge*. We can think about a typology of incompleteness in knowledge based on three dimensions of variability. These are *the nature of the incompleteness*, the *type of knowledge that is incomplete*, and whether the incompleteness is represented in a *quantitative or qualitative* manner.

With regard to the nature of the incompleteness, in the simplest case we may have a variable or variables that have a defined set of possible values or hypotheses from which the true value is known to be drawn. We refer to this as *simple uncertainty*. We can also have *uncertainty about the number of variables* needed in a model, i.e. about the model complexity. Finally we can also have cases where the agent knows that a variable is of an unexperienced class, i.e. there is *novelty*. This can include cases where the variables are continuous but where the observation models for a class are quite confident and do not generalise well to some new observation. The type of knowledge that is incomplete may vary enormously. Four simple types that cover a variety of cases include contingent knowledge about the current world *state*, *structural* knowledge about the universal relationships between variables, knowledge consisting of *predictions* of action outcomes or events, and knowledge about their *causes*. Finally there is a question about whether the representation is qualitative or quantitative. In qualitative representations of gaps or uncertainty we have a set of possible values for the variable, or a statement that the variable value is unknown, or knowledge that there may be many variables that are unmodelled. In quantitative representations we will have some kind of scalar values attached to hypotheses (e.g. is this a cup or mug) or statements (such as whether there is novelty or not), and in our case these will typically be probabilities. Note that by a *quantitative gap* or *quantitative uncertainty* we do not mean that the underlying space for the variable is continuous or discrete, but instead that the way the incompleteness is represented involves an expression of preference for one hypothesis or statement versus another.

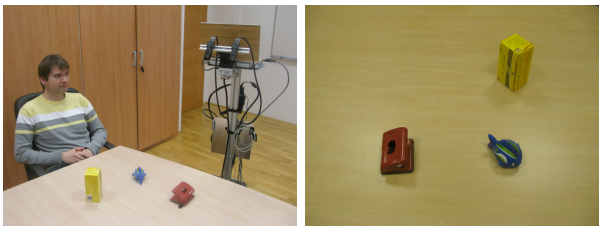
In this paper we deal with filling qualitative gaps, qualitative uncertainty in state, quantitative uncertainty about structural knowledge, and novel states. We provide empirical proof that our approach works and illustrate different aspects of it through two robot systems we have implemented. We call these Dora, and George (Figs. 1 and 2). We provide links to



(a) The Dora platform: a P3 mobile robot base with a custom-built superstructure and different sensors.

(b) Visualisation of Dora's map of a partially explored environment. Coloured disks denote *place* nodes (colour indicates segmentation into different rooms, `area0,1,2`). Small green circles represent opportunities for spatial exploration (*placeholders*). Red nodes indicate places where doorways are located.

Fig. 1. Dora the Explorer: a robot system for goal-driven spatial exploration.



(a) Scenario setup. (b) Observed scene.

Fig. 2. George scenario: continuous interactive learning of visual properties.

videos of these systems running in the real world[1], and analyse their behaviour on larger amounts of data using simulations. We now describe Dora and George in more detail to give concrete examples that will run through the paper.

Dora is able to explore an office environment, choosing between filling different types of incompleteness in her maps of the environment. Dora illustrates the architecture, the representations of gaps in spatial knowledge, the use of epistemic states in goal setting and action planning, and in the use of our motivation system. In Dora's case the current incomplete knowledge she can model and fill can be seen by reference to Fig. 1(b). Here we can see a visualisation of a map that Dora has built after a partial tour of an office environment. The map consists of a graph where the nodes (which we call *places*) are partitioned into areas by landmarks, such as doors. Dora has representations of two kinds of incompleteness. She represents unexplored regions of space, by maintaining a set of hypothesised places, which we call *placeholders*. These are depicted in Fig. 1(b) as small unfilled circles with numeric labels. This is uncertainty about how many variables are needed to model the space. Second, Dora has the ability to categorise areas into categories such as office, kitchen, coffee room and corridor. In Fig. 1(b) it can be seen that none of the areas currently have known categories. This is simple state uncertainty, as Dora knows a certain number of types of area, and cannot represent or reason about novel area types. During the autonomous part of the mapping process Dora will choose the order in which to reduce these different kinds of incompleteness. To map unexplored areas by adding nodes to the topological map she will conduct laser based mapping while visiting the hypothesised placeholders. To categorise a room she will search for objects that indicate its category, e.g. kitchens typically contain objects such as milk cartons and cups, and offices objects such as bookshelves and journals.

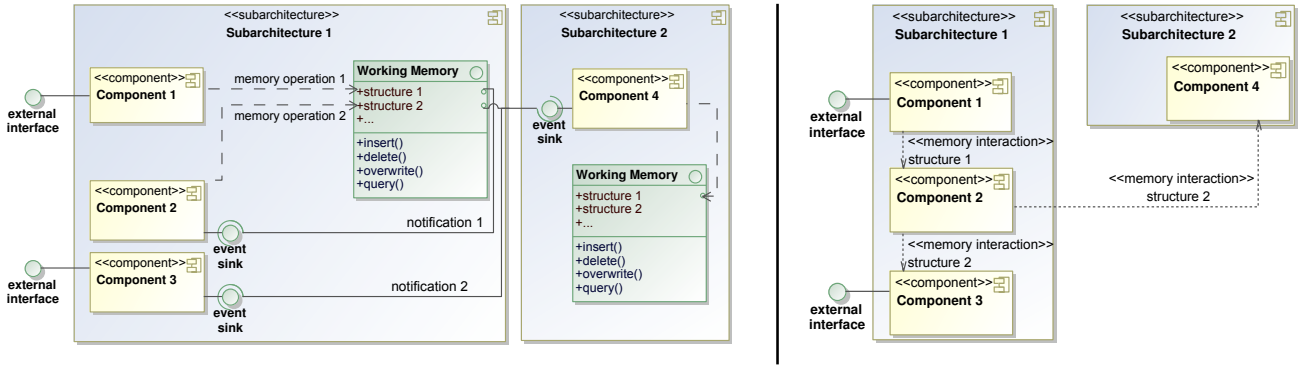
George is a system that converses with a human to reduce incompleteness it has about the properties of objects on a table top. George illustrates the way we represent uncertainty and incompleteness in models of the structural relationships between visual information and linguistic descriptions of objects. What visual properties, for example, make an object round or square, vs. red or yellow? George has representations of the uncertainty it has as to which sub-spaces of a set of visual features are associated with particular adjectives. This is a type of structural uncertainty. George can learn from a tutor, but crucially he can also decide which questions to ask in order to fill a particular gap he has identified. A typical scene during George's learning is depicted in Fig. 2. A typical dialogue snippet might be:

G: Which colour is the elongated object?
H: The elongated object is yellow.
G: OK.
G: Is the square object blue?
H: No it is not blue. It is red.
G: OK.

During this dialogue the robot and the human reason about each others beliefs, what they know and don't know, and how to establish common understanding. This is type of state uncertainty, since the robot can only model the human as having one of a known set of beliefs. In the dialogue each agent makes references to objects that they understand will be distinguishing to the other agent, such as referring to the elongated object. More importantly George asks questions which are prompted by detection of gaps such as state novelty. He asks: *"Which colour is ..?"* when he realises that the colour is one he hasn't experienced before. When he is simply uncertain about which of a number of colour classes is present he asks instead whether the object has the most likely colour class: *Is the object blue?*. Both George and Dora also have mechanisms for doing non-monotonic inference or learning. George can unlearn erroneous representations of colour and shape, and in Dora's case she can withdraw support for inferences about room category, or the partition of her map.

The rest of the paper is structured as follows. In Section II we describe the architectural model. Section III describes the model that connects information from multiple modalities, and how we have engineered those representations to explicitly capture uncertainty and incompleteness in the amodal model. In Section IV covers representations of space, cross-modal relations, epistemic goals, and epistemic action effects, all as used in Dora and/or George. In Section V we describe our approach to goal management, and finally in Sections VI and VII we describe the Dora and George systems and present an experimental analysis their behaviour.

[1]Available at `http://cogx.eu/`



(a) In CAS, there are components, which run in parallel, asynchronously updating shared structures on a common working memory. They can also take input from sensors or give output to actuators. Subarchitectures are coupled to make an overall system.

(b) A simplified illustration of the interaction patterns mediated through the working memories focusing on information flow from sources to sinks.

Fig. 3. Building systems with CAS. The form on the right is used as a short hand in the later system diagram for Dora.

II. An Architecture for Multi-Modal Processing

In this section we describe the basic architecture we employ, which we call CAS (CoSy Architecture Schema) [1]. We refer to it as a schema because it actually defines a space of specific architectures, we refer to the schema when talk about this space, and to an architecture when we mean a specific architecture employed in a particular robot system. The schema is essentially a distributed working memory model, where representations are linked within and across the working memories, and are updated asynchronously and in parallel. The key idea is that it replaces a single world model (still prevalent in robotics) with multiple, linked world models, enabling it to work in the face of inconsistent evidence, uncertainty and change. The decomposition into working memories groups processes that commonly share information, and is typically by sensory modality. So that in Dora and George we build separate subsystems (called *subarchitectures*) for vision, communication and spatial understanding. As we shall see in Sections III and IV each subarchitecture contains representations which explicitly capture uncertainty and incompleteness. The system overall can reason about this uncertainty or incompleteness and plan how to act so as to fill that knoweldge gap, perhaps by employing information in another modality. We now describe the key aspects of CAS relevant to this paper.

A. Subarchitecture Design

1) Components: Our schema starts on the level of a collection of processing components (Fig. 3(a)). Every component is concurrently active, allowing them to process in parallel. We do not specify any constraints on the contents of components: they could have behave like a node in a connectionist network, an activity in a behaviour-based system, or an entire function in a functionally-composed system. Components can take input either directly from sensors, or from working memory. They can also directly control actuators in the manner of closed loop controllers, or initiate fixed action patterns. Components can have processing triggered by the appearance of certain information on the shared working memory, and can modify structures on that memory. Components may also have their own private (i.e. component-internal) memory. Components are typically designed around two archetypes: managed and unmanaged. *Unmanaged components* are low-latency processes that run all the time, regardless of overall system state. *Managed components* by contrast are typically computationally expensive processes, which only run when there is a demand for their services. These components are only run when a particular configuration of information is present on working memory.

2) Shared Working Memories: Rather than exchange information directly, processing components are connected to a *shared working memory* (Fig. 3(a)). The contents of the working memory are solely composed of the outputs of processing components. Each working memory is connected to all other working memories in the system. This allows components to exchange information across subarchitectures. In our implementation of CAS the communication method between the working memory and the components determines the efficiency of the model. But for now let us consider simply that the schema itself allows reading and writing to working memories, and transfer of information between them.

This use of shared working memories is particularly well suited to the *collaborative refinement of shared structures*. In this approach to information processing, a number of components use the data available to them to incrementally update an entry on working memory. In this manner the results of processing done by one component can restrict the processing options available to the others in an informed way. As all components are active in parallel, the collective total processing effort (i.e. the amount of work done by all the components in solving a problem) may be reduced by sharing information in this way. This feature turns out to be a very powerful aspect of the schema.

B. System Design Practices with CAS

While a system could be composed of a single subarchitecture, we intend that there should typically be several subarchitectures in operation. In the integrated systems we describe in this paper we have about four subarchitectures. The architecture makes no assumptions about whether system decomposition should be predominantly according to behaviour

or information processing function. What is important is that the decomposition groups components that commonly exchange information via shared structures. One of the main benefits of having distributed working memories is that the working memory can act as a filter for its local components, only letting them become aware of events elsewhere in the system when necessary.

In the systems described here (Dora and George) we have separate subarchitectures for vision, linguistic communication, and spatial understanding. To have coherent global action, however, the system benefits from linking these separate models. In other words there are data structures, or symbols on one blackboard, that are related to those on another. For example the visual system might have a symbol corresponding to a blue cup sitting on a table, and the communication system might have a symbol corresponding to a mention of a blue thing by a human. To make sense of the human's words the robot has to decide whether these two symbols are connected, or whether the human is referring to some other object (perhaps another blue object in the scene). An important question is what mechanisms can be used to link these efficiently? We call this symbol linking problem the *binding problem* and we describe it in the next section. In particular we will talk about how we can solve the binding problem in a way that allows us to represent uncertainty in which symbols should be bound to which, and also gaps in the robot's overall picture of the world right now. Binding essentially involves creating new symbols that refer back to each of the modality specific symbols. We refer to the representations that are created by binding as multi-modal representations. At the highest level of abstraction, however, binding produces an essentially amodal model of the robot's world.

III. Modelling multi-modal beliefs

So far we have described an architecture capable of supporting processing on groups of modality specific representations. High-level cognitive capabilities must generally operate on high level (i.e. abstract) representations that collect information from multiple modalities. This requirement raises the double issue of (1) how these high-level representations can be reliably generated from low-level sensory data, and (2) how information arising from different subsystems can be efficiently fused into unified multi-modal structures.

We present here a new approach to multi-modal information binding [2], [3], based on a Bayesian framework. The approach is implemented in a specific subarchitecture in our systems called the *binder* [4]. The binder is directly connected to all subsystems in the architecture. It serves as a central hub for the information gathered about entities currently perceived in the environment. The data structures included in the binder are inherently probabilistic. Each property or information bit pertaining to an entity is given a probability value, reflecting the confidence level of the subsystem. This enables the system to deal with varying levels of noise and uncertainty, which are unavoidable for most sensory-motor processes.

Based on the data structures made available in this repository, the binding algorithm seeks to "merge" or unify the perceptual inputs arising from the various subsystems, by checking whether their respective features correlate with each other. The probability of these correlations are encoded in a Bayesian network. This Bayesian network might for instance express a high compatibility between the haptic feature "shape: cylindrical" and the visual feature "object: mug" (since most mugs are cylindrical), but a very low compatibility between the features "shape: cylindrical" and "object: ball".

The resulting multi-modal information structure is called a *belief* in our terminology. The task of the binder is to decide which proxies from different modalities belong to the same real-world entity, and should therefore be merged into a belief. The outcome of this process is a joint probability distribution over possible beliefs. These beliefs integrate in a compact representation of all the information included in the perceptual inputs. They can therefore be directly used by the deliberative processes for planning, reasoning and learning.

A. Representations

The three central data structures manipulated by the binder are **proxies**, **unions** and **beliefs** (also see Fig. 4(a)).

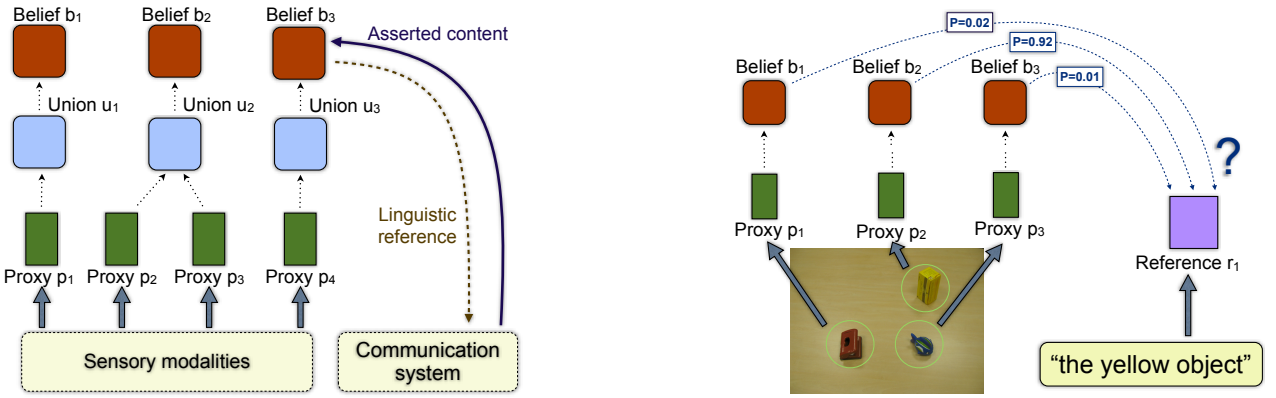
1) Proxies: A mid-level, uni-modal representation of a given entity in the environment. Proxies are inserted onto the binder by the various subsystems included in the architecture.

A proxy is essentially defined as a multivariate probabilistic distribution over a set of features. The distributions included in the proxy can be either discrete (as for categorical knowledge) or continuous (as for real-valued measures).

2) Unions: A mid-level, multi-modal representation of an entity, constructed by merging one or more proxies. Just like proxies, unions are also represented as a multivariate probabilistic distribution over possible features. Unions are essentially a transitional layer between proxies and beliefs.

3) Beliefs: A high-level, amodal representation of an entity in the environment. Beliefs are generally build on top of unions, but they are expressed in an amodal format and encode additional information related to the specific situation and perspective in which the belief was formed, such as its *spatio-temporal frame*, its *epistemic status* and its *saliency value*:

- The spatio-temporal frame is defined according to a spatial model (set of possible "places" in the environment), a temporal model (points on a continuous temporal interval), and possibly a perspective on these two models from the viewpoint of a particular agent.
- The epistemic status of a belief (or subpart of a belief) can be either private, attributed or shared. Private beliefs are beliefs which are internal to the agent, while attributed beliefs are beliefs an agent ascribes to another agent (e.g. A believes that B believes X). Shared beliefs are beliefs which are part of the common ground for all agents.
- Finally, the salience is a multivariate density function $\Re^n \rightarrow [0, 1]$, where each variable defines a particular, real-valued saliency measure. It provides an estimate of the "importance" or quality of standing out of a particular entity relative to neighboring ones [5]. The salience is used to drive the attentional behaviour of the agent by specifying which entitites are currently in focus.



(a) Construction of multi-modal beliefs.

(b) Reference resolution for the expression "the yellow object".

Fig. 4. Multi-modal information binding: belief construction (left) and application in a reference resolution task (right).

Beliefs are indexed via an unique identifier, which allows us to keep track of the whole development history of a particular belief. Beliefs can also be connected with each other using relational structures of arbitrary complexity.

To account for this rich representation, beliefs are formalised according to a *belief model*, which is a mathematical structure defining a space of possible belief instances.

B. Binding algorithm

To be able to create beliefs out of proxies, the binder must decide for each pair of proxies arising from distinct subsystems, whether they should be bound into a single union, or fork in two separate unions. The decision algorithm for this task is based on a well-known technique from probabilistic data fusion, called the *Independent Likelihood Pool* (ILP) [6]. Using the ILP, we are able to compute the likelihood of every possible binding of proxies, and use this estimate as a basis for constructing the beliefs. The multivariate probability distribution contained in the belief is a linear function of the feature distributions included in the proxies and the correlations between these.

A Bayesian network encodes all possible feature correlations as conditional dependencies. The encoded features may be discrete or continuous. The (normalised) product of these correlations over the complete feature set provides an useful estimate of the "internal consistency" of the constructed belief – a belief with incompatible features will have a near-zero probability, while a belief with highly correlated features will be associated with a high probability.

C. Referencing and updating beliefs

The beliefs are high-level symbolic representations available for the whole cognitive architecture. As such, they provide an unified model of the environment which can be efficiently used when interacting with the human user. An important aspect of this is *reference resolution*: how to connect linguistic expressions such as "this box" or "the ball on the floor" to the corresponding beliefs about entities in the environment.

Reference resolution is performed using the same core mechanisms as for binding – a Bayesian network specifies the correlations between the linguistic constraints of the referring expressions and the belief features (in particular, the entity saliency and associated categorical knowledge). The resolution process yields a probability distribution over alternative referents (see Fig. 4(b) for an example), which is then retrieved by the communication subsystem for further interpretation.

In addition to simple reference, the interaction with a human user can also provide *new* content to the beliefs, as in cross-modal learning scenarios. Via (linguistic) communication, the human user can thus directly extend or modify the robot's current beliefs, in a top-down manner, without altering the incoming proxies. If this new information conflicts with existing perceptual knowledge, the agent can decide to trigger a clarification request to resolve the conflict.

An utterance such as "This is yellow" illustrates these two complementary mechanisms. First, the linguistic expression "this" must be resolved to a particular entity in the environment. Since "this" is a (proximal) deictic, the resolution is performed on basis of the saliency measures. In the absence of any other constraint, the most salient entity is simply selected and retrieved. Second, the utterance not only refers to an existing entity in the environment, but it also provides new information about it – namely that is is yellow. This asserted content must therefore inserted into the robot's beliefs. This is realised by selecting the belief pertaining to the referred-to entity and incorporating the new, attributed information into its content representation.

D. Implementation

The outlined approach has been fully implemented as a separate subsystem in the cognitive architecture. It includes a central working memory where proxies can be inserted, modified or deleted. The belief set is automatically updated to reflect the incoming information. A GUI allows the user to monitor at runtime the binder behaviour.

The Bayesian network encoding the feature correlations can be either manually specified, or learned using various machine learning techniques (see section VII).

IV. Representations to support self-extension

This section explains how different domains of knowledge are represented in our system. We present three types of representations: representations of space, primarily related to the Dora scenario; cross-modal representations used in the George scenario; and representations of epistemic state and action effects used by planning. Each representation focuses on structuring and abstracting what is known, but also on representing uncertainty and knowledge gaps explicitly.

A. Representations of space

Spatial knowledge constitutes a fundamental component of the knowledge base of a mobile agent, such as Dora, and many functionalities directly depend on the structure of the spatial knowledge representation. These include spatial localisation, navigation, wayfinding and autonomous exploration, but also understanding and exploiting semantics associated with space, human-like conceptualisation and categorisation of and reasoning about spatial units and their relations, human-robot communication, action planning, object finding and visual servoing, and finally storing and recalling episodic memories.

In our system, spatial knowledge is represented in multiple layers, at different levels of abstraction, from low-level sensory input to high level conceptual symbols. Moreover, continuous space is discretised into a finite number of spatial units. The abstraction and discretisation processes is one of the most important abstracting steps in representing spatial knowledge as it allows to make the representation compact, tractable and robust to changes that occur in the world. Discretisation drastically reduces the number of states that have to be considered, e.g., during the planning process and serves as a basis for higher level conceptualisation.

The representation is designed for representing complex, cross-modal, spatial knowledge that is inherently uncertain and dynamic. Our primary assumption is that spatial knowledge should be represented only as accurately as it is required to provide all the necessary functionality of the system. This keeps the complexity of the representation under control, makes the knowledge more robust to dynamic changes and substantially reduces the effort required to synchronise the representation with the environment. Additionally, uncertainties are associated with the represented symbols and gaps in spatial knowledge are explicitly modelled.

Fig. 5 gives a general overview of the structure of the representation. It is sub-divided into layers of specific representations. We distinguish between four layers which focus on different aspects of the world, abstraction levels of the spatial knowledge and different spatial scales. Moreover, each layer defines its own spatial entities and the way the agent's position in the world is represented. At the lowest abstraction level we have the sensory layer which maintains an accurate representation of the robot's immediate environment extracted directly from the robot's sensory input. Higher, we have the place and categorical layers. The place layer provides fundamental discretisation of the continuous space explored by the robot into a set of distinct places. The categorical layer focuses on low-level, long-term categorical models of

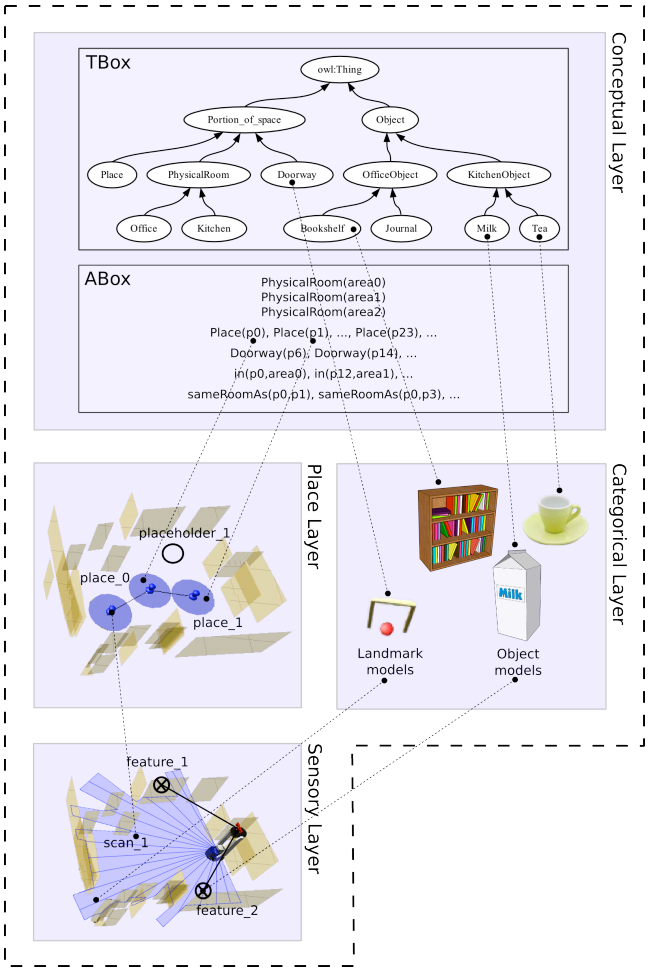


Fig. 5. The layered structure of the spatial representation. The position of each layer within the representation corresponds to the level of abstraction of the spatial knowledge. The ABox in the conceptual layer corresponds to the example in Fig. 1(b) on page 2.

the robot's sensory information. Finally, at the top, we have the conceptual layer, which associates human concepts (e.g., object or room category) with the categorical models in the categorical layer and groups places into human-compatible spatial segments such as rooms.

The following paragraphs provide additional details about each of the layers and their instantiations within our system. The system provides only an initial instantiation of the representation that validates correctness and usefulness of the knowledge structure within an integrated cognitive system. At the same time, as it will be mentioned below, some of the underlying algorithms do not adhere fully to the principles behind the representation. For a detailed theoretical discussion on those principles and optimal implementations, we refer the reader to [7].

1) Sensory Layer: In the sensory layer, a detailed model of the robot's immediate environment is represented based on direct sensory input as well as data fusion over space around the robot. The sensory layer stores low-level features and landmarks extracted from the sensory input together with their exact position. Measures of uncertainty are also included in this representation. Landmarks beyond a certain distance

are forgotten and replaced by new information. Thus, the representation in the sensory layer is akin to a sliding window with robocentric and up-to-date direct perceptual information.

The representation in the sensory layer helps to maintain stable and accurate information about the robot's relative movements. Moreover, it allows for maintaining and tracking the position of various features while they are nearby. Finally, the sensory layer provides the low level robotic movement systems with data for deriving basic control laws, e.g., for obstacle avoidance or visual servoing.

In the current implementation, the sensory layer is maintained by two subsystems: a metric SLAM algorithm [8] that builds a global metric map of the environment and an Active Visual Search (AVS) component which represents hypotheses about objects found in the environment using a local grid map. The SLAM algorithm explicitly represents the uncertainty associated with the pose of the robot and the location of all landmarks using a multivariate Gaussian distribution encoded using a state vector and a covariance matrix [9], [8]. At the same time, the AVS component maintains hypotheses about existence of an object of a specific category at a specific location using a probabilistic grid representation [10]. The probabilistic grid representation is shaped based on multiple cues about object location one of which is the presence of obstacles in the SLAM map. This is the prior on which the AVS algorithm determines the next best viewpoint based on a randomized art-gallery algorithm [11].

The existence of the global metric map violates some of the assumptions behind the proposed representation; however, it is only used internally. In the future instantiations, the allocentric SLAM algorithm will be replaced by a robocentric method [12], [13], [14]. Here, in order to verify the correctness of such concept, we constrain access to the metric map from other components of the system, exposing only local and relative (with respect to the robot) metric information – with the exception of the navigation system that still uses the allocentric SLAM algorithm.

2) Place Layer: The place layer is responsible for the fundamental, bottom-up discretisation of continuous space. In the place layer, the world is represented as a collection of basic spatial entities called *places* as well as their spatial relations. The aim of this representation is not to represent the world as accurately as possible, but at the level of accuracy sufficient for performing required actions and robust localisation despite uncertainty and dynamic variations.

Besides places, the place layer also defines paths between them. The semantic significance of a path between two places is the possibility of moving directly between one and the other. In addition, the place layer explicitly represents *gaps in knowledge about explored space*. Space that has not yet been explored by the robot has no places in it. Therefore, tentative places are generated, which the robot would probably uncover if it moved in a certain direction. These hypothetical places allow for reasoning about unknown space, and for planning and executing exploratory activities. They are annotated as *placeholders* to keep them apart from ordinary, actual places, but are otherwise identically represented and interconnected. For an illustrative example of several places and placeholders identified during spatial exploration, see Fig. 1(b) on page 2.

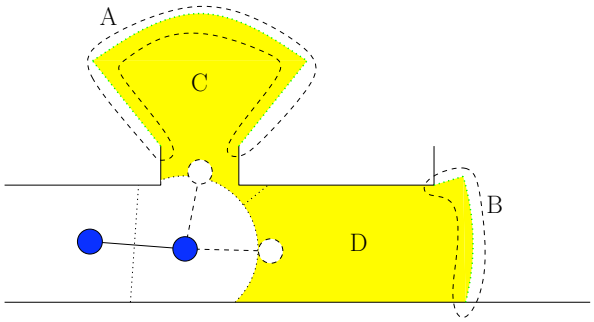


Fig. 6. Placeholder creation. Dashed circles are hypotheses, each representing one placeholder. A and B are frontier length estimates, C and D are coverage estimates for the respective placeholders.

Two quantitative measures are associated with each placeholder providing an estimate of information gain related to each exploration task. These are used by the motivation system, described later in Section V on page 11. The measures used are the *coverage estimate* (CE) and the *frontier length estimate* (FLE), cf. Fig. 6. The former is obtained by measuring the free space visible from the current node and not near to any existing node, and assigning it to the closest hypothesis. This heuristically estimates the number of new places that would result from exploring that direction. The FLE is analogously extracted from the length of the border to unknown space. By prioritising these two measures differently, the motivation mechanism can produce different exploratory behaviours.

3) Categorical Layer: The categorical layer contains long-term, low-level representations of categorical models of the robot's sensory information. The knowledge represented in this layer is not specific to any particular location in the environment. Instead, it represents a general long-term knowledge about the world at the sensory level. For instance, this is the layer where models of landmarks or objects are defined in terms of low-level features. The position of this layer in the spatial representation reflects the assumption that the ability to categorise and group sensory observations is the most fundamental one and can be performed in a feed-forward manner without any need for higher-level feedback from cognitive processes.

The categorical models stored in this layer give rise to concepts utilised by higher-level layers. In many cases complex models are required that can only be inferred from training data samples. In case of models that correspond to human concepts, they can be learnt in a supervised fashion, using a top-down supervision signal.

In our system, the categorical layer was realised through visual categorical models of objects employed by the AVS component and a simple door detection algorithm used as a landmark model. The AVS component uses the object recognition method proposed in [15] and the models associated with object classes reside in the categorical layer. However, using only this algorithm does not provide object pose with the uncertainty associated with it and is not robust to cases where two objects appears similar from a certain viewpoint. Therefore, a natural extension to this procedure which estimates the pose and class of objects is also implemented [10] .

Additionally, in our experiments, we employed appearance and geometry-based models of place categories [16]. Although, currently not being used in the Dora scenario, those models constitute a part of the categorical layer.

4) Conceptual Layer: The conceptual layer provides a symbolic ontological representation of space that makes use of human-compatible concepts. The taxonomy of the spatial concepts and properties of spatial entities, as well as the instances of these concepts are linked to the lower levels of the spatial model. This associates semantic interpretations with the low-level models and can be used to specify which properties are meaningful, e.g., from the point of view of human-robot interaction. The main purpose of the conceptual layer is to represent a segmentation of the environment into rooms. Moreover it provides human-compatible concepts for these rooms based on the objects they contain, and it can supply default assumptions about which kinds of objects are likely to be found in which kinds of rooms.

The representation underlying the conceptual map is an OWL-DL ontology[2], consisting of a taxonomy of concepts (*TBox*) and the knowledge about individuals in the domain (*ABox*), cf. Fig. 5 on page 6, cf. [17]. Here is an example of a *concept definition* in the current implementation which defines a kitchen as a room that contains at least two typical objects:

$$\mathsf{Kitchen} \equiv \mathsf{Room} \sqcap \geq 2\mathsf{contains}.\mathsf{KitchenObject}$$

Besides the usual inferences performed by the OWL-DL reasoner, namely *subsumption checking* for concepts in the TBox (i.e., establishing subclass/superclass relations between concepts) and *instance checking* for ABox members (i.e., inferring which concepts an individual instantiates), an additional *rule engine* is used to maintain a symbolic model of space under *incomplete* and *changing* information.

The discrete places from the place layer and their adjacency are the main pieces of knowledge that constitute the input for that reasoning. One, it maintains a representation that groups places into rooms. Furthermore, using observations (visually detected objects, appearance- and geometry-based room categories) it can infer human-compatible concepts for a room, and raise expectations about which other kinds of objects are proto-typically likely to be present. The ongoing construction of the conceptual map is potentially nonmonotonic. The overall room organisation may be revised on the basis of new observations. The further association between room concepts and salient, proto-typical object types is established through the "locations" table of the OpenMind Indoor Common Sense[3] database by Honda Research Institute USA Inc.

In the current implementation, the conceptual layer can be used to determine *knowledge gaps in the categorisation of rooms.* It is considered a gap in knowledge if for a given room (i.e., an instance of PhysicalRoom) its basic level category is unknown. This is assumed to be the case if no more specific concept than PhysicalRoom (i.e., Office or Kitchen, cf. Fig. 5 on page 6) can be inferred for the individual. This knowledge gap persists until the robot has gathered enough evidence (i.e., contained objects) for inferring a subconcept.

[2]http://www.w3.org/TR/owl-guide/
[3]http://openmind.hri-us.com/

B. Representations of epistemic state and action effects

Decisions about what actions to perform next are not pre-programmed in our robot, but are made by the *planning* subarchitecture. In this section, we describe how knowledge and knowledge-producing actions are modelled such that the planner can reason about how knowledge gaps can be filled. Planning systems traditionally use representations based on propositional logic. Most notably, the classic STRIPS formalism and its modern descendent PDDL are based on such a propositional representation. The representation we use for the planning system on our robot, however, is based on the SAS$^+$ formalism [18]. Here, instead of propositions, we use multi-valued state variables (MVSVs) v, each with an associated domain $vdom(v)$ describing the set of possible values $x \in vdom(v)$ that v may assume. A state is a function s associating variables with values from their domain. In recent years, SAS$^+$ has been shown to enable powerful reasoning techniques in planning algorithms and systems based on SAS$^+$ now dominate the International Planning Competition. For the modelling needs of our robot applications, we have developed the SAS$^+$-based modelling language MAPL [19].

For robotic planning, MAPL provides, in addition to the computational advantages, several representational ones. Firstly, it stays close to the feature/value model used by other subarchitectures of our robot. In particular, the mapping between binder states and planning states is greatly simplified: Roughly, each feature f of a union u in a belief model is mapped onto a state variable $f(u)$. For example, if the belief model describes that a room has been categorised as a kitchen by attributing the feature *areaclass : kitchen* to a union u, this would correspond to an assignment *areaclass(u) = kitchen* in a planning state.

The main reason for using an SAS$^+$-based representation is that we can employ it to explicitly model knowledge and gaps in knowledge, so that the planner can efficiently reason about them. To this end, we must relax the assumption that in a state s all variables v have a value x. Instead, we accept states that are only partially defined, i.e., where some variables are undefined or "unknown". Conversely, we also need to represent future states in which gaps will have been filled. By nature, we can not know in advance which value a variable v will assume then, but we can nevertheless exploit the knowledge that, e.g., after executing a sensing action the value of v will be "known". To this end, we use so-called *Kval* variables Kval_v with $vdom(\mathsf{Kval}_v) = \top, \bot$. With *Kval* variables we can also model the epistemic effects of sensing actions. For example, the action of running a room categorisation algorithm in a room is modelled in MAPL as follows:

```
(:sensor categorise_room
 :agent (?a - agent)
 :parameters (?r - room ?loc - place)
 :precondition
    (= (pos ?a) ?loc)
    (contains ?r ?loc)
 :effect (Kval ?a (areaclass ?r))
)
```

In words, this action model describes that an agent can sense the area class of a room, i.e. its being a kitchen,

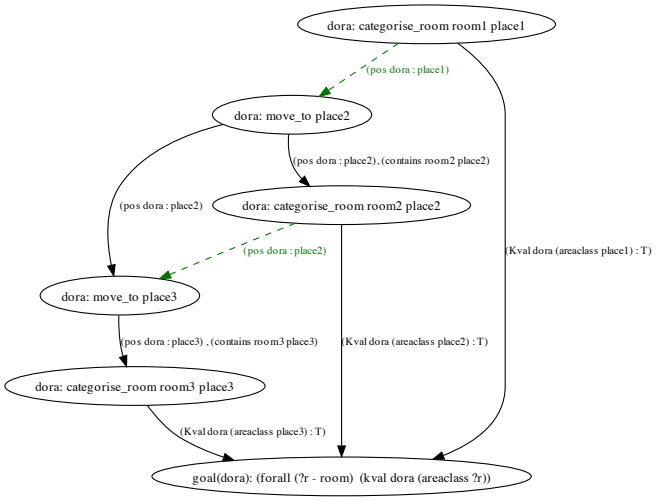


Fig. 7. A plan using sensory actions to satisfy epistemic goals.

office or hallway, once the agent is at a place that belongs to the room in question. At planning time, the outcome of observing *areaclass(r)* is yet unknown, therefore the effect of `categorise_room(r,loc)` is formally described as $Kval_{areaclass(r)} = \top$.

Of course, *Kval* variables can appear in goal formulae as well, so that we can conveniently express *epistemic goals*, i.e. goals concerned with closing knowledge gap. Goal formulae can contain expressions in first-order logic, in particular conditionals and quantifiers, so that we can give the robot goals like "categorise all rooms and explore all currently known places" , which would correspond to $\forall loc.Kval_{areaclass(loc)} = \top \wedge \forall place.explored(place)$.

Interestingly, due to the use of a quantified formula the goal will be re-evaluated repeatedly during the continual planning process, i.e. the planner will autonomously adapt its plan to explore and categorise newly discovered places and rooms. A (slightly simplified) example of a plan using sensing actions that satisfy epistemic goals is given in Fig. 7.

In the George scenario and in our next instantiation of Dora, information will not only be obtained by sensing, but also through interaction with humans. To plan for such multiagent interactions the robot must also reason about the knowledge of the other agents. We can express nested beliefs using MVSVs as well, e.g., "the robot R believes that human H believes that object o is a pen" is modelled as $B^{R,H}_{type(o)} = pen$. Knowledge gaps may arise in several variants when nested beliefs are used, depending on which agent is ignorant of the other's belief. Again, with MVSVs we can represent the differences succinctly using agent-specific "unknown" symbols. Consider, e.g., the difference between the statements "R knows that H does not know the location of the cornflakes" ($Kval^{R,H}_{pos(cornflakes)} = \bot^H$) and "R does not know if H knows the location of the cornflakes" (($Kval^{R,H}_{pos(cornflakes)} = \bot^H$). Just as sensing actions are modelled using standard *Kval* variables, we can use nested *Kval* variables to describe *speech acts*. In particular, we can describe *wh-questions* and answers to them ("where", "what colour", etc.) by modelling the appropriate nested belief effects. (Note: the planner was not used for dialogue planning in the George system as presented in this paper, but will be in its next instantiation).

C. Representations for cross-modal learning

Cross-modal learning plays an important role in a self-extending system. It enables the system to, based on interaction with the environment and people, extend its current knowledge by learning about the relationships between symbols and features that arise from the interpretation of different modalities. It involves processing of information from multiple modalities, which have to be adequately represented. One modality may exploit information from another to update its current representations, or several modalities together may be used to form representations of a certain concept. In this subsection we focus on the former type of interaction between modalities and present the representations that are used for continuous learning of basic visual concepts in a dialogue with a human. While Section III describes the formation of belief models, which supervise the learning in the visual domain, this subsection focuses on representations that are being updated in this continuous learning process. All these principles are integrated and demonstrated in the George scenario described in Section VII.

1) Representations for visual concepts: The visual concepts are represented as generative models, probability density functions (pdf) over the feature space, and are constructed in online fashion from new observations. In particular, we apply the online Kernel Density Estimator (oKDE) [20] to construct these models. The oKDE estimates the probability density functions by a mixture of Gaussians, is able to adapt using only a single data-point at a time, automatically adjusts its complexity and does not assume specific requirements on the target distribution. A particularly important feature of the oKDE is that is allows adaptation from the positive as well as negative examples [21]. The continuous learning proceeds by extracting visual data in a form of a highdimensional features (e.g., multiple 1D features relating to shape, texture, color and intensity of the observed object) and oKDE is used to estimate the pdf in this high-dimensional feature space. However, concepts such as *color red* reside only within lower dimensional subspace spanned only by features that relate to color (and not texture or shape). Therefore, during online learning, this subspace has to be identified to provide best performance. This is achieved by determining for a set of mutually exclusive concepts (e.g., colors green, blue, orange, etc.) the subspace which minimizes the overlap of the corresponding distributions. The overlap between the distributions is measured using the Hellinger distance as described in [22]. Therefore, during online operation, a multivariate generative model is continually maintained for each of the visual concepts and for mutually exclusive sets of concepts the feature subspace is continually being determined. The set of mutually exclusive concepts can then be used to construct a Bayesian classifier in the recognition phase, when the robot is generating a description of a particular object in terms of its color, shape, etc. However, since the system is operating in an online manner, the closed-world assumption can not be assumed; at every step the system

should take into account also the probability of the "unknown model" as described in the following.

2) Accounting for unknown model: While maintaining good models of the visual concepts and being able to adapt those models is crucial for the robots online operation, the ability to detect gaps in the knowledge presented by these models is equally important. Generally speaking the robot collects the visual information about its environment as follows. First it determines a region in an image which contains the interesting information, then it "segments" that region and extracts the feature values z from which it later builds models of objects, concepts, etc. The visual information may be ambiguous by itself, and segmentation may not always be successful. We will assume that some measure of how well the segmentation was carried out exists and we will denote it by $s \in [0, 1]$. High values of s (around one) mean high confidence that a good observation z was obtained, while low values relate to low confidence.

Let $m \in \{m_k, m_u\}$ denote two possible events: (i) the observation came from an existing internal model m_k, and (ii) the observation came from an unknown model m_u. We define the knowledge model as a probability of observation z, given the confidence score s:

$$p(z|s) = p(z|m_k, s)p(m_k|s) + p(z|m_u, s)p(m_u|s). \quad (1)$$

The function $p(z|m_k, s)$ is the probability of explaining z given that z comes from one of the learnt models, $p(m_k|s)$ is the a priori probability of any learnt model given the observer's score s. The function $p(z|m_u, s)$ is the probability of z corresponding to the unknown model, and $p(m_u|s)$ is the probability of the model "unknown" given the score s.

Assume that the robot has learnt K separate alternative internal models $M = \{M_i\}_{i=1:K}$ from previous observations. The probability $p(z|m_k, s)$ can then be further decomposed in terms of these K models,

$$p(z|m_k, s) = \sum_{i=1}^{K} p(z|M_i, m_k, s)p(M_i|m_k, s). \quad (2)$$

If we define the "unknown" model by M_0 and set $p(z|m_u, s) = p(z|M_0, m_u, s)p(M_0|m_u, s)$, then (1) becomes

$$p(z|s) = p(m_k|s) \sum_{i=1}^{K} p(z|M_i, m_k, s)p(M_i|m_k, s) + p(m_u|s)p(z|M_0, m_u, s)p(M_0|m_u, s). \quad (3)$$

Note that the "unknown model", M_0, accounts for a poor classification, by which we mean that none of the learnt models supports the observation z strongly enough. We assume that the probability of this event is uniformly distributed over the feature space, which means that we can define the likelihood of model M_0, given observation z by a uniform distribution, i.e., $p(z|M_0, m_u, s) = \mathcal{U}(z)$. Note also, that the only possible unknown model comes from the class M_0, therefore $p(M_0|m_u, s) = 1$.

The observation z can be classified into the class M_i which maximizes the a posteriori probability (AP). The a posteriori probability of a class M_i is calculated as

$$p(M_i|z, s) = \frac{p(z|M_i, m, s)p(M_i|m, s)p(m|s)}{p(z|s)}, \quad (4)$$

where $m = m_k$ for $i \in [1, K]$ and $m = m_u$ for $i = 0$.

In our implementations, the distribution of each i-th alternative of the known model $p(z|M_i, m_k, s)$ is continuously updated by the oKDE [20], while the a priori probability $p(M_i|m_k, s)$ for each model is calculated from the frequency at which each of the alternative classes M_i, $i > 0$, has been observed. The a priori probability of an unknown model (and implicitly of a known model), $p(m_u|s)$ is assumed non-stationary in that it changes with time. The following function decreases the "unknown" class probability with increasing number of observations N:

$$p(m_u|s) = e^{-0.5(\frac{N}{\sigma_N})^2}, \quad (5)$$

where σ_N is a user specified parameter that specifies how the robot's internal confidence about learned models changes with time.

With above definitions, the knowledge model is completely defined and allows discovery of knowledge gaps. They can be discovered through inspection of the AP distribution. In particular, we can distinguish two general cases:

- The observation z can be best explained by the unkown model, which indicates the gap in the knowledge; the observation should most probably be modeled with a model, which has not yet been learned.
- The a priori probability of the model that best explains the observation is low, which indicates that the classification is very uncertain and that the current model can not provide a reliable result.

3) Illustrative example: For a better visualization of the knowledge update and gap discovery we will restrict our example to a one-dimensional case. Fig. 8 illustrates detection and filling of knowledge gaps for three cases (feature values) denoted by the circle, the diamond, and the square. The plots in the left column depict the models and the recognition at a particular step in the learning process, while the right column depicts the situation after the system has updated these models

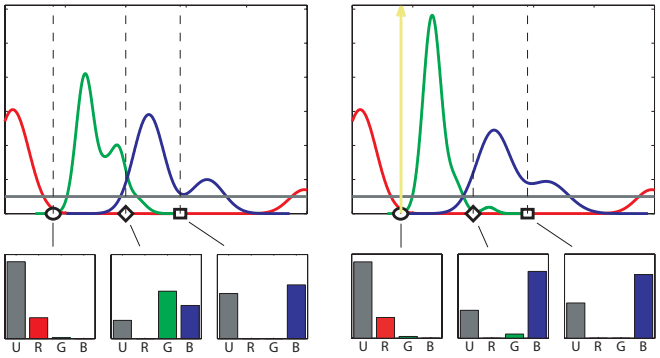


Fig. 8. Example of detecting the knowledge gaps and updating the 1D KDE representations. Top row: probability distributions for three colours (red, green, blue lines) and unknown model (gray line) in 1D feature space. Bottom row: a posteriori probabilities for the unknown model (U) and three colours (R, G, B) for three feature values denoted by the circle, the diamond and the square. Left column: before updates, right column: after updates.

considering the detected knowledge gaps and the answers from the tutor.

Consider a scene similar to that presented in Fig. 2. Let us assume that the circle in Fig. 8 represents the yellow object and that the yellow colour has not been presented to the robot before. Therefore, the corresponding model for colour yellow has not yet been learned and the feature value obtained from the segmented yellow object fails in a not yet modeled area. This value is thus best explained by the "unknown model", which has the highest a posteriori probability. The robot detects this gap in his knowledge and asks the tutor "*Which colour is this object?*", and after the tutor provides the requested information, the robot initializes a model for yellow colour. However, since only one sample does not suffice to build a reliable representation, the yellow colour will only be able to be recognized after some additional yellow objects are observed.

The feature value denoted by a diamond in Fig. 8 is best explained by a green model, however this recognition is not very reliable, therefore the robot asks the tutor: "*Is this object green?*" to verify its belief. After the tutor replies "*No. It is blue.*", the robot first unlearns the representation of green and updates the representation of blue. The corrected representations, depicted in the pdfs in the right column in Fig. 8, then enable the correct recognition as indicated by the second bar plot in the right column of the Fig. 8.

The last case denoted by the square shows another example of non-reliable recognition, which triggers the additional clarification question to the tutor: "*Is this object blue?*" After the robot gets a positive answer, it updates the representation of blue, which increases the probability of the recognition.

V. Goal management: choosing between different epistemic goals

In the previous sections the focus was very much on the *representation* of knowledge gaps and on the *understanding* of knowledge limitations. In this section we discuss a generic framework to *generate* and *manage* epistemic goals that correpsond to knowledge gaps in these representation. We propose a *goal generation and management framework (GGM)* that enables the robot to decide *which* gaps in its representation to eliminate *when* and generate appropriate behaviour to self-extend its knowlegde.

We have built on the work of [23], to produce the design illustrated in Figure 9. This design is a general framework, or schema, for an architecture for goal generation and management that tackles the mentioned issues. It specifies a collection of interacting elements which must be included in any instantiation of the framework, although the precise details of the instantiation will inevitably vary between instances. The elements of the framework are described in more detail below.

At the bottom of the framework, a system's drives are encoded as multiple *goal generators*. These are concurrently active processes which monitor the system's state (both the external world and internal representations) and produce *goals* to satisfy the system's drives. Generators can also remove previously generated goals if they are judged to no longer

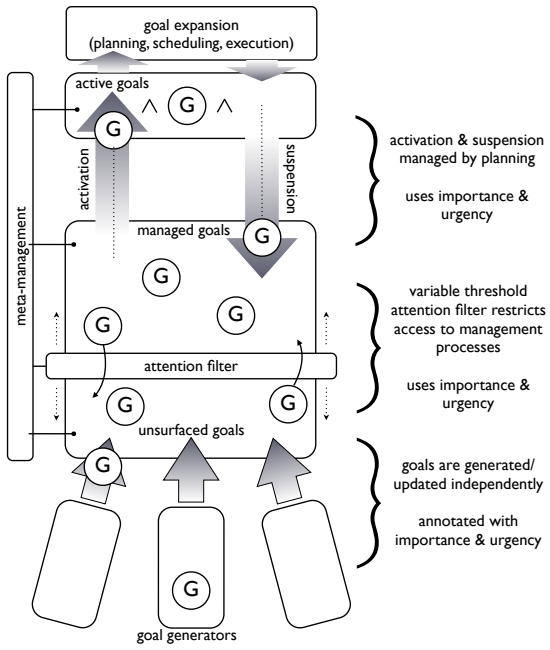


Fig. 9. The goal generation and management framework.

be appropriate. In this manner we can say that the system's drives are encoded in the goal generators (either explicitly or implicitly). We work from the assumption that as a goal passes up through the framework from a generator and influences a system's behaviour, it is inspected by processes of greater and greater computational complexity. Therefore the lower strata of the framework exist to protect these processes (and thus overall system resources) from having to consider more goals than is necessary (where this could be a contingent judgement). The main mechanism in the framework for protecting the management processes is the *attention filter*. This is a coarse barrier which uses simple, fast processing to let some some goals through to the management mechanisms whilst blocking others. Goals which make it through this filter are described as *surfaced*, thus the goals which fail to pass the filter are referred to as *unsurfaced*. A collection of management processes determine which of the surfaced goals should be combined to serve as the goals being actively pursued by the system. If a goal is selected in this way we describe it as *activated*. If a goal is removed from the set of goals being pursued by the system we refer to it as *suspended*.

In order to fulfil their roles, the filtering and management processes require information on which to base their decisions. Following the original work [23], the framework requires that goal generators annotate each goal with a description of the goal's *importance* and *urgency*, and keep these descriptions up to date as long as the goal exists. Importance should reflect the significance of the goal to the agent (as motivated by the related drive). Urgency should reflect the necessity of achieving the goal sooner rather than later. As we shall see later, producing importance and urgency descriptions for use in such a framework is a problem in itself. In addition to these descriptions, the framework allows the management processes

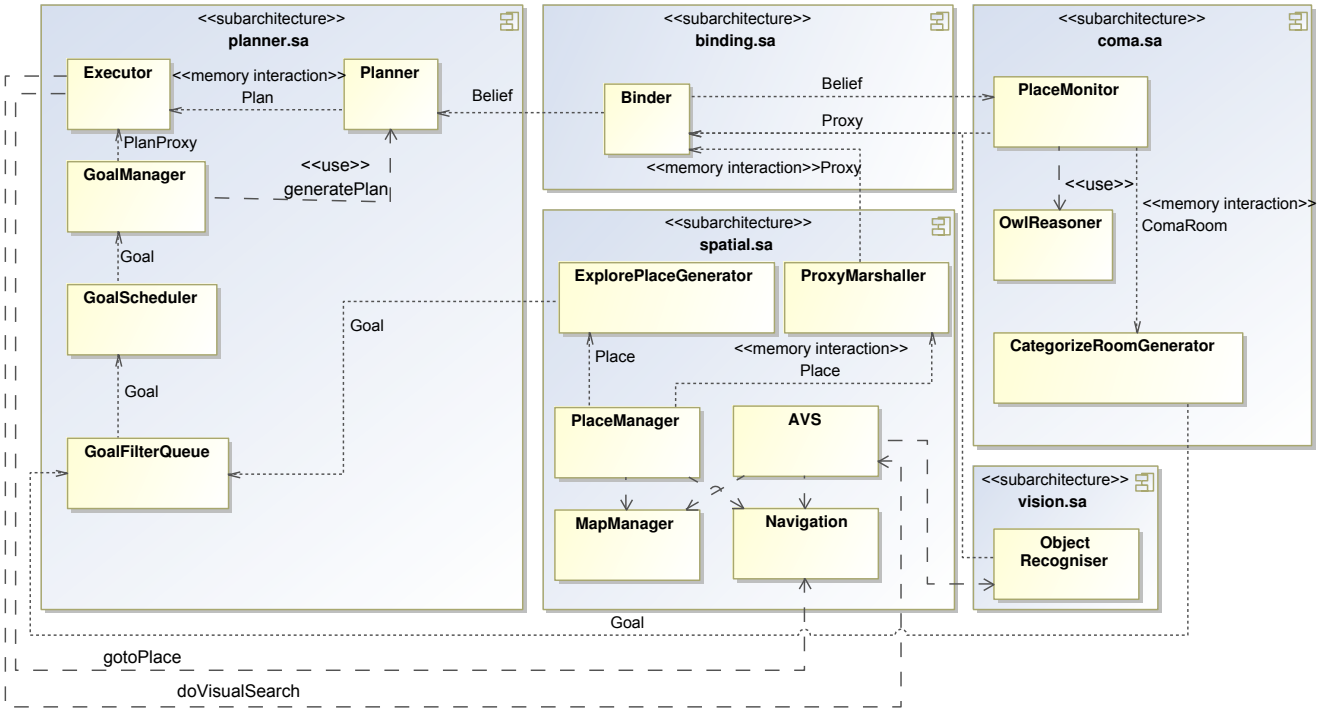


Fig. 10. Dora system architecture. For clarity, all memory interactions are not depicted as information flow mediated through working memories in the subarchitectures but as directed dotted connections of sources and sinks. The dashed lines represent synchronous request-reply calls to components by mutually modifying memory structures.

to use whatever approaches are required to select and maintain a set of active goals. Perhaps the minimum requirements on these processes is the ability to check whether a goal, or collection of goals, can be achieved (thus positing planning as a goal activation, as well as achievement, mechanism).

The GGM is currently implemented as one of the core concepts in our exploring robot Dora (cf. Sec. VI). In this system we derive the importance of a goal from an estimated *information gain* computed for the epistemic goals. In brief, the information gain for achieving the goal of exploring a yet unexplored place is derived from the measures shown in Fig. 6. The information gain of categorizing a room is similarly designed, assuming that a categorising bigger rooms yields more information. The GGM continuously monitors these measures of information gain and relates it to the *costs* to actually achieve this goal acquired by asking the planner. We are currently not employing a notion of *urgency* in our implementation as the robot's drives are not prioritised so far.

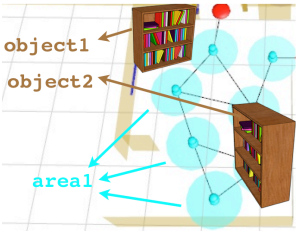
The GGM in cooperation with planning implements *action selection and execution* in our systems, allowing the robots to expose effective and efficient self-extending behaviour.

VI. Dora the Explorer

The current implementation of Dora is focused on spatial representations and two types of knowledge gaps that give rise to epistemic goals to fill these gaps: explore a place and categorise a room. Dora's system architecture is composed of five of the subarchitectures discussed earlier in this paper running all on one Laptop computer on the autonomous robot, cf. Fig. 1(a). The composition is sketched in Fig. 10. The diagram is adopted from UML 2.0 specification and illustrates the information flow between components, and also across subarchitectures. Most of the flow is realised by interactions with the working memories in an event-driven manner as proposed by CAS in Sec. II. For clarity and readability the interactions are not shown as connections to the respective memories but linking sources and sinks of information directly following the schema pictured in Fig. 3(b). The diagram does however not include all components. We focus here on those that are required to understand the architecture facilitating self-understanding and -extension, disregarding those that can be seen only as support services and sensor abstraction.

The application domain of Dora is also reflected in this architecture when we compare it to the architecture of George: spatial.sa is only relevant for a mobile robot and the reasoning about spatial concepts implemented in coma.sa is likewise specific to the scenario (cf. Sec. IV-A4). Furthermore, Dora is studied as our first system that employs goal management and planning to choose and pursue epistemic goals.

vision.sa only plays a minor role in this system. The ObjectRecogniser component (based on the FERNS detector [24]) detects objects visually when triggered by AVS. The ProxyMarshalling is a mediator component selecting spatial information to be presented as Proxy to the binder. The PlaceMonitor in coma.sa fulfills a similar purpose. As can be seen in the figure, binding.sa in Dora as in George is working on a unified representation of beliefs and proxies only, allowing a simple transformation of information in the planning domain required by the Planner. As can been seen as well, the goal management scheme of motivation in Dora is implemented as part of planner.sa. Epistemic goals are created from observing knowledge gaps in the specific representations



(a) Dora has found 2 bookshelves.

Bookshelf(object1)
Bookshelf(object2)
in(object1,area1)
in(object2,area1)

→ Office(area1)

(b) New facts stored in the ABox.

Fig. 11. Continuing the example in Fig. 1(b) on page 2: based on the presence of 2 OfficeObject instances the DL reasoner infers that area1 instantiates Office.

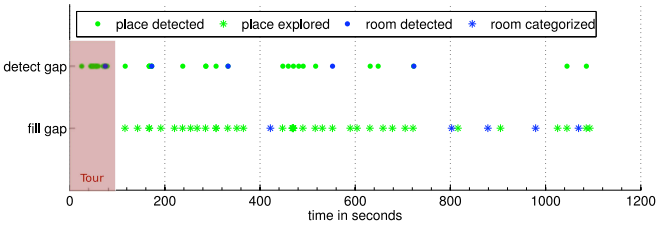


Fig. 12. Exemplary course of action for filling knowledge gaps in a real run.

in other SA's working memories as discussed in Sec. IV-A2 and IV-A4. These goals are filtered, scheduled, and planned for following the principles introduced in Sec. V. The Executor eventually executes and monitors actions by triggering either the navigation to explore a place or the AVS component (cf. Sec. IV-A1) which autonomously interacts with other components to visually search for objects that allow the OwlReasoner to derive conceptual knowledge about rooms (cf. Sec. IV-A4).

Fig. 10 on the previous page also illustrates our decomposition strategy. The only representations that are currently exchanged across borders of subarchitecture are information related to binding (cf. Sec. III) and the epistemic goals corresponding to the knowledge gaps.

A. *Dora running*

A prototypical run with the current Dora implementation unfolds as follows:

1) Dora starts from scratch without any knowledge about the specific environment she is operating in.
2) Optionally, Dora can be given a short tour by a human instructor to create an initial representations already containing some knowledge gaps. Fig. 1(b) on page 2 shows such a partially known environment.
3) Dora autonomously explores her environment having drives to self-extend with respect to two types of knowledge gaps: unexplored places as they have been defined in the place layer and yet uncategorised rooms as defined in conceptual layer of the spatial representations. In the example in Fig. 1(b) on page 2, the rooms area0, area1, area2 give rise to room categorisation drives, whereas the different placeholders lead to exploration goals. Note that a number of placeholders (notably the ones labelled "8(7)", "7(16)", and "20(19)") are in space that will later be segmented into new rooms, which then, in turn, will also need to be categorised.
4) A room categorization goal is considered satisfied when a more specific concept can be inferred for a PhysicalRoom instance in the ABox of the conceptual map layer, cf. Fig. 11. An exploration goal is satisfied if the robot either turns the placeholder into a real place or discards it, because it turned out as a false hypothesis.

The two types of gaps are created and monitored by the components PlaceManager@spatial.sa and PlaceMonitor@coma.sa, respectively. Fig. 10 illustrates how these components submit hypotheses about ComaRoom (a detected but not yet categorised room) and Place (a detected but not yet explored place) to their working memories. From these gaps epistemic goals are generated by the goal generators ExplorePlaceGenerator and CategoriseRoomGenerator, respectively. Thus, a number of goals is always present in the system corresponding to these gaps. Dora's behaviour is driven by the selection of a subset of these goals by goal generation and management (GGM, cf. Sec. V) and the execution of actions according to generated plans to achieve these goals.

Fig. 12 visualises an exemplary run of Dora illustrating the course of action she takes with respect to self-extending for the two types of knowledge gaps. The x-axis shows the time in seconds since the beginning of the run. The figure thus indicates the time when a new gaps are detected (upper line) and the time, when a particular epistemic goal has been accomplished to fill a gap in the knowledge. This particular run comprised an initial tour taking Dora from the corridor (the long room in the centre of Fig. 1(b) on page 2) to the room in the upper-right corner in that figure. It can be seen in Fig. 12 that she is passively detecting gaps in her knowledge in the phase labelled "Tour", but not yet autonomously extending it. Only after the tour Dora interleaves categorisation of rooms to fill gaps with the exploration of new places. A video[4] of the real robot operating in an office environment can support comprehension of this illustration and provide the reader with a better understanding of Dora's generated behaviour.

Taking a closer look on the actual processes in the system, we can see how the interaction between component works. Fig. 13 on the following page pictures the activity that the robot goes through from the detection of a knowledge gap to its filling. The example illustrated in the figure is corresponding to "explore place" only, but the structure is similar for "categorise room". It starts with spatial.sa hypothesising a new place and thus generating a Place in the working memory that is marked as being hypothetical. This generation triggers binding and ExplorePlaceGenerator to create a Belief about this place and an epistemic Goal to explore this place. After the motivation-related components have filtered and scheduled the generated goal, the planner is triggered to generate a plan to actually achieve it. The Executor then executes the actions of the plan. One action will be to navigate towards the placeholder which will – in this example – make it explored. This update is again propagated through the working memory, resulting in the goal to be removed and the belief being updated asynchronously.

[4] http://cogx.eu/results/dora/

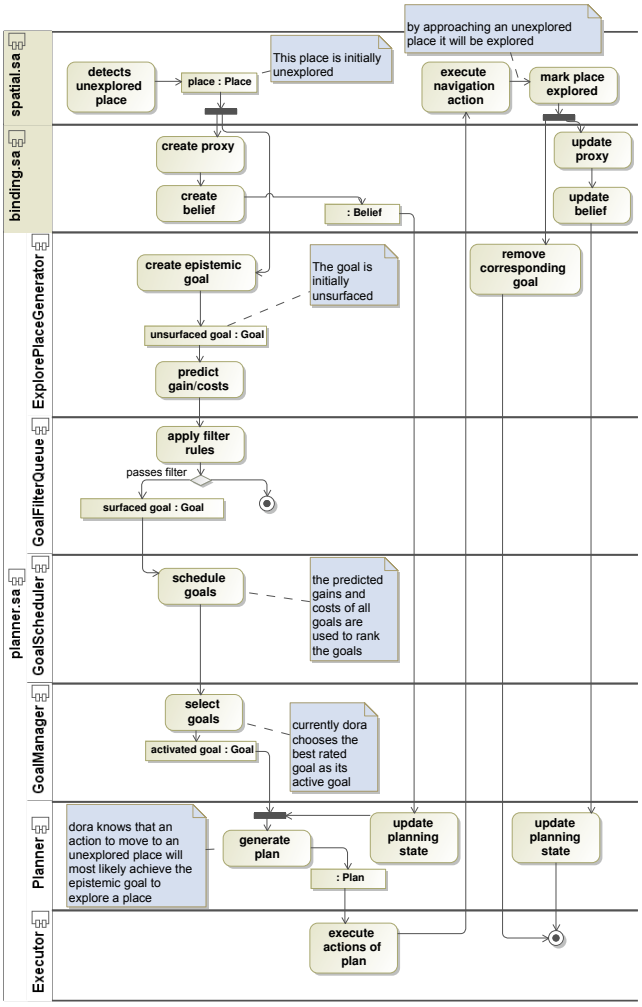


Fig. 13. Activity diagram illustrating the "path" of a knowledge gap from its generation to its filling.

In this paper we present two experiments, each studying slightly different aspects of the Dora scenario. The first experiment focuses on the spatial representation with an emphasis on the nonmonotonic reasoning about space in the conceptual layer. The second one focuses on the goal management and the benefits of having such a management framework in the system. For sake of reproducibility and focus on specific aspects these experiments are carried out in simulation. Our simulation framework transparently substitutes the sensors and actuators, still allowing to run the core system unmodified and in a situated way with sensing-actuation loops closed. The simulator operates using the floor plan of one of the real environments in which Dora operates (cf. Fig. 14).

B. Experiment 1: Spatial Representation

One consequence of the uncertainty and partiality of the observations Dora is dealing with is that the map building process is nonmonotonic. Structural and conceptual abstractions may need to be reconsidered in the light of new evidence acquired during the active exploration. In this experiment we assess the accuracy and appropriateness of our nonmonotonically built spatial representation as the robot keeps exploring.

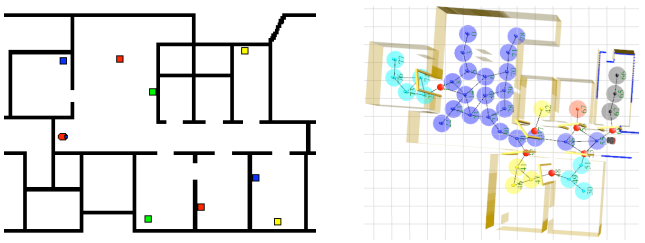


Fig. 14. Stage simulation model used in the experiments (l) and screenshots of the visualisation tool acquired during one of the three experiments (r).

Setup: The map consisted of eight rooms: a corridor, a terminal room, a lab, two offices, two restrooms, and a printer room, cf. Fig. 14. This constitutes the ground truth for our tests of the accuracy of the room maintenance. The robot was ordered to perform an autonomous exploration, which means that only placeholder exploration goals were considered by the motivation system. To evaluate the coverage that this exploration yields, we determined a gold standard of 60 Place nodes to be generated in order to fully, densely and optimally cover the simulated environment. We achieved this by manually steering the robot to yield an optimal coverage, staying close to walls and move in narrow, parallel lanes.

We performed three runs with the robot in different starting positions, each time with an empty map. Each run was cut-off after 30 minutes. The robot was then manually controlled to take the shortest route back to its starting position.

For the evaluation, the system logged the state of its ABox each time a new room was created, or an existing one was deleted. This subsumes cases in which rooms are split or merged. At each such step, the generated map was compared to the ground truth for the room representation and to the gold standard for Place node coverage. The first room instance to cover part of a ground-truth room is counted as *true positive (TP)*. If that room instance extends into a second room, it is counted as TP only once, and once as a *false positive (FP)*. Each additional room instance inside a ground-truth room is also counted as FP. *False negatives (FN)* are ground-truth rooms for which no room instance exists. Using these measures, precision P, recall R and the balanced f-score F for the room maintenance are as follows: $P = \mid TP \mid/(\mid TP \mid+\mid FP \mid)$, $R = \mid TP \mid/(\mid TP \mid+\mid FN \mid)$, $R = 2 \times ((P \times R)/(P+R))$. We compute a normalised value for coverage using $coverage = \mid nodes \mid/60$.

Results: Fig. 15 on the next page shows the development of the relevant measures during the three experimental runs. As can be seen, the accuracy (balanced f-score) of the representation is monotonically increasing towards a high end result (0.8, 0.79 and 0.93, resp.). The increases and decreases in precision during the individual runs are due to the introduction and retraction of false room instances. Recall can be interpreted as coverage in terms of room instances. After 30 minutes the exploration algorithm yielded a relatively high recall value (0.75, 0.75 and 0.875, resp.), i.e., most of the rooms had been visited. A recurring problem here was that the two smallest rooms were often only entered by a few decimetres. This was enough to consider the corresponding Placeholder to be explored, but not enough to create an additional Place

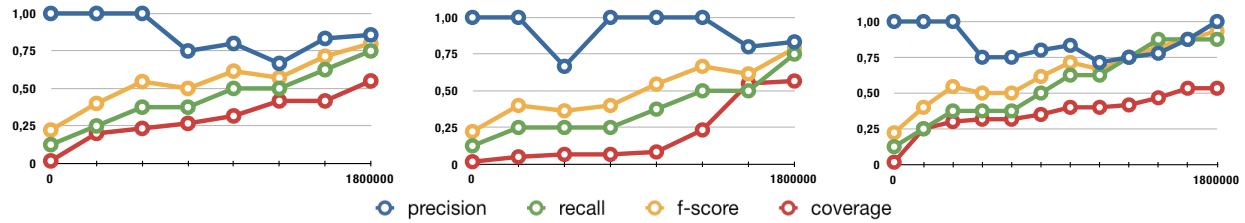


Fig. 15. Plots for precision, recall, balanced f-score and coverage of each of the three experimental runs. The Y-axis shows the normalised values for precision, recall, balanced f-score, and coverage (0–1). The X-axis is time, in milliseconds.

node beyond the doorway – which would have been the prerequisite for room instance creation. The node coverage that the algorithm achieved after 30 minutes (33, 34, 32 out of 60, respectively) can be attributed partly to the 30-minutes cut-off of the experiment, and partly to the exploration strategy which goes for high information gain Placeholder first. These tend to be in the middle of a room rather than close to its walls.

C. Experiment 2: Goal Management

As discussed in Sec. IV-B and V, we have two alternatives to express Dora's drives. First, we can explicitly use quantifiers to create on conjunctive goal for the overall system to explore all places and categorise all rooms and rely on the replanning ability of the continual planning (cf. Sec. IV-B). We term this system setup *Conjunct Goal Set (CGS)*. The proposed alternative is to make use of the goal generation and management approach and let it select and schedule the individual goals. Our hypothesis is that (i) the effort for planning is reduced as we chunk the problem into smaller pieces, making it tractable if it comes to more complex problems, and (ii) goal management is a powerful and simple means to encode domain knowledge into the behaviour generation in Dora. We refer to this second setup as *Managed Goal Set (MGS)*.

Setup: For this study we restricted the space to be explored five rooms and the corridor (being the right part of Fig. 1(b) on page 2 without the large room). The ultimate goal in this setup for the robot is to categorise all rooms using the two typical objects placed in each of the five rooms. The objects describe one of three categories according to the OpenMind Indoor Common Sense Database (room, office, and kitchen), allowing the conceptual mapping SA to categorise these rooms.

A single run starts with a short tour through the corridor. Then Dora is switched to autonomous mode and starts acting in response to her goals. Fig. 12 on page 13 is generated from one of these runs including the tour and the categorisation of five rooms. In total we ran the system 15 times: 8 in MGS configuration and 7 in CGS. A run for the CGS configuration was defined as complete when the conjunctive goal was achieved (i.e., no places left unexplored and no rooms uncategorised). The MGS configuration was said to be complete when no more surfaced goals remained.

Results: As part of our experiments that are fully detailed in [25] we were interested in the effect the GGM approach has on the complexity of problems to be solved by planning. So me measured the time Dora spend planning in the runs for the two different setups. These measures are summarised in Table I.

TABLE I
PLANNING TIME MEASURES (ALL IN SECONDS).

	CGS	**MGS**
avg. time per planning call	0.621 s	0.292 s
avg. time spent on planning	48.843 s	8.858 s

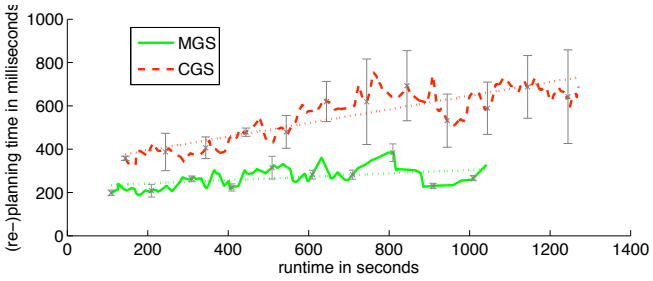


Fig. 16. Averaged planning time during a system run.

The differences between the averaged timings taken for the two configurations are statistically significant with $p < 0.0001$ in Mann-Whitney testing for all measures shown in the table.

As the first row of the table indicates, there is a significant difference between the average time taken by a single call to the planner. A call occurs either when the goal management activates a new goal or when replanning is triggered by a state change. Planning calls in CGS take more than twice the time compared to MGS. This is due to the higher complexity of the planning problems in the CGS configuration (it is planning for the conjunction of all epistemic goals rather than a single goal). If we look at the average time spent on planning in total per run (second row in Table I) the difference is more prominent. This is due to the fact that in the CGS configuration the planner is triggered more often: 79.0 times on average, compared to 31.1 times for the MGS configuration. This is because the longer plan lengths required in CGS are more likely to be affected by state changes and thus require more frequent replanning.

Figure 16 shows how the complexity of planning problems evolves as the system is running. It depicts the length of single planner calls against the runtime of the system. For comparability, this plot has been created from a partial set of all runs (five of each configuration) containing only those in which Dora successfully categorised all five rooms. The planning time is averaged at discrete time steps across all the runs of each setup. The error bars indicate the standard error in averaging. From this figure it is apparent that, in agreement with the data in Table I, less planning effort is required in MGS compared to CGS. It can also be seen that the progression over runtime is different in the two cases. While the trend, indicated

by a linear fitting shown as a dotted line in Fig. 16, is a shallowly included line for MGS, a steeper increase in average planning time can be seen for CGS. This steeper increase can be associated with the increasing size of the planning problems the CGS configuration faces as Dora's knowledge increases: planning for all possible goals over a larger and larger state becomes increasingly difficult. This underpins our hypothesis that with a suitable mechanism for goal selection we can tackled the challenge of increasingly complex environments and correspondingly high numbers of knowledge gaps in our representations.

VII. George: curiosity driven cross modal learning

The George scenario has been designed to demonstrate, monitor, and show progress on the development of the integrated system for *learning the association between visual features of an object and its linguistically expressed properties*. The main goal is, therefore, to integrate the developed vision routines, learning and recognition competencies, dialogue capabilities, as well as different kinds of representations and belief models in an overall system.

A. Scenario setup and example script

The robot operates in a table-top scenario, which involves a robot and a human tutor (see Fig. 2(a)). The robot is asked to recognize and describe the objects in the scene (in terms of their properties like colour and shape). There are a single or several objects (i.e., up to five) in the scene (but still, with limited occlusion). The human positions new objects on the table and removes the objects from the table while being involved in a dialogue with the robot. At the beginning the robot does not have any representation of object properties, therefore he fails to recognize the objects and has to learn. To begin with, the tutor guides the learning and teaches the robot about the objects. After a while, the robot takes the initiative and tries to detect the ignorance and to learn autonomously, or asks the tutor for assistance when necessary. The tutor can supervise the learning and correct the robot when necessary; the robot is able to correct erroneously learned representations. The robot establishes the transparency and verbalizes its knowledge and knowledge gaps, as well as intended actions. In a dialogue with the tutor, the robot keeps extending and improving the knowledge. The tutor can also ask questions about the scene, and the robot is able to answer (and keeps answering better and better). At the end, the representations are rich enough to accomplish the task - to correctly describe the initial scene.

Two main types of learning are present in the George scenario, which differ on where the motivation for learning update comes from. In tutor driven learning the learning process is initiated by the human teacher, while in the tutor assisted learning, the learning step is triggered by the robot.

Tutor driven learning is suitable during the initial stages, when the robot has to be given information, which is used to reliably initiate (and extend) visual concepts. Consider a scene with a single object present:

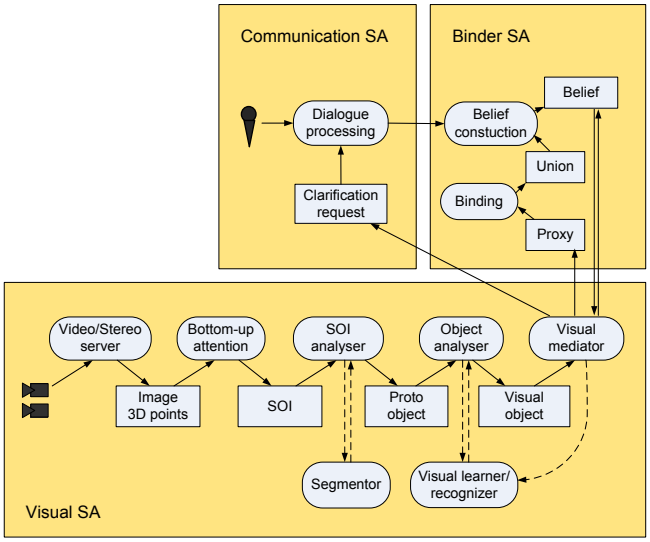


Fig. 17. Architecture of the George system.

H: Do you know what this is?
G: No.
H: This is a red object.
G: Let me see. OK.

Since at the beginning George doesn't have any representation of visual concepts, he can't answer the question. After he gets the information, he can first initiate and later sequentially update the corresponding information.

After a number of such learning steps, the acquired models become more reliable and can be used to reference the objects. Therefore, there can be several objects in the scene, as in Fig. 2, and George can talk about them:

H: What colour is the elongated object?
G: It is yellow.

When the models are reliable enough, George can take initiative and try to learn without being told to. In this curiosity driven learning George can pose the question to the tutor, when he is able to detect the object in the scene, but he is not certain about his recognition. As described in Section IV-C in such *tutor assisted* learning there are two general cases of detecting uncertainty and knowledge gaps. If the robot can not associate the detected object with any of the previously learned models, it considers this as a gap in his knowledge and asks the tutor to provide information:

R: Which colour is this object?
H: It is yellow.
R. OK.

The robot is now able to initialize the model for yellow and, after the robot observes a few additional yellow objects, which make the model of yellow reliable enough, he will be able to recognize the yellow colour.

In the second case, the robot is able to associate the object with a particular model, however the recognition is not very reliable. Therefore, the robot asks the tutor for clarification:

R: Is this red?
H: No. This is yellow.
R. OK.

After the robot receives the answer from the tutor, he corrects (unlearns) the representation of the concept of red and updates the representation of yellow and makes these two representations more reliable.

In such mixed initiative dialogue George continuously improves the representations and learns the reliable models of basic visual concepts. After a while George can successfully recognize the acquired concepts and provide reliable answers:

H: Do you know what this is?
G: It is a blue object.
H: What shape is the red object?
G: It is elongated.

B. System architecture and processing pipeline

The George system is composed of three subarchitectures: *Binder SA*, *Communications SA* and *Visual SA*, as depicted in Fig. 17. The components of visual subsystem (SA) can be divided in three distinct layers: the quantitative layer, the qualitative layer and the mediative layer.

The quantitative layer processes the visual scene as a whole and implements one or more *bottom-up* visual attention mechanisms. A bottom-up attention mechanism tries to identify regions in the scene that might be interesting for further visual processing. George has currently one such mechanism, which uses the stereo 3D point cloud provided by *stereo reconstruction component* to extract the dominant planes and the things sticking out from those planes. Those sticking-out parts form spherical 3D spaces of interest (SOIs). The *SOI Analizer* component validates the SOIs and, if deemed interesting (SOI persistence, stability, size, etc.), upgrades them to *proto-objects* adding information that is needed for the qualitative processing (e. g. segmentation mask).

The qualitative layer processes each interesting scene part (object) individually, focusing on qualitative properties. After the extraction of the visual attributes (Visual Learner-recognizer), like color and shape, the *Object Analyzer* upgrades the proto-objects to *visual objects*. Visual objects encapsulate all the information available within Visual SA and are the final modal representations of the perceived entities in the scene. Also, the learning of visual attributes is performed on this layer.

The main purpose of *the mediative layer* is to exchange information about the perceived entities with other modalities. This is usually not done directly, but via specialised a-modal subarchitectures like the Binder SA (Section III). The *Visual Mediator component* adapts and forwards the modal information about objects to the binder (each visual object is represented by a dedicated proxy in the binder). The component also monitors beliefs for possible learning opportunities, which result in modal learning actions. Another important functionality of the mediator is to formulate and forward clarification motivations in the case of missing or ambiguous modal information. Currently, these motivations are directly intercepted by Communication SA, which synthesizes a question about the certain object property.

Now, let us describe the processing pipeline on one illustrative example. We will describe in more detail what happens after the human places several objects in the scene (see Fig. 2) and refers to the only elongated object in the scene (the yellow tea box) by asserting "*H: The elongated object is yellow.*".

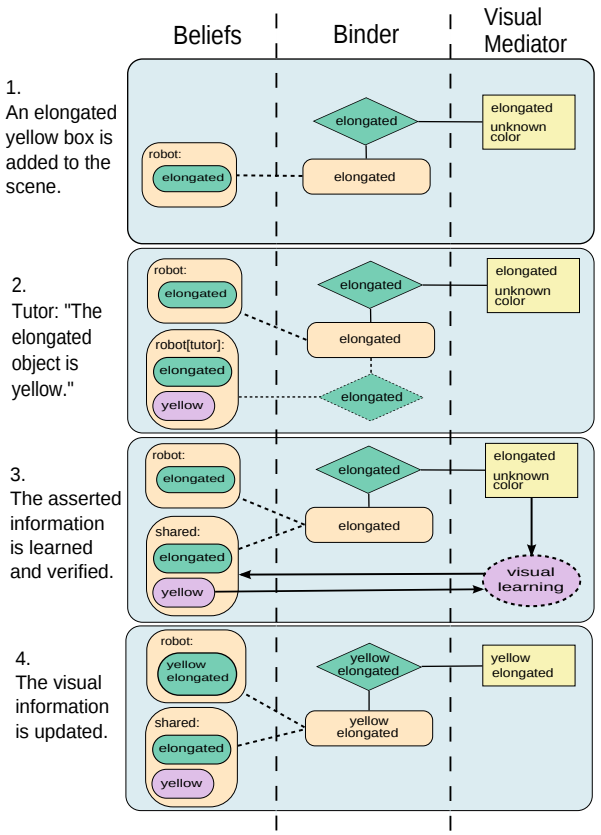


Fig. 18. Example of processing pipeline. The green color represents restrictive information, while the violet color denotes asssertive information. Only the beliefs and other data structures pertaining to the yellow tea box are shown.

In Visual SA the tea box is represented by a *SOI* on the quantitative layer, a *proto-object* on the qualitative layer and a *visual object* on the mediative layer. Let us assume that the *Visual Learner-recognizer* has recognized the object as of elongated shape, but has completely failed to recognize the color. In the binder this results in a one-proxy union with the binding features giving the highest probability to the elongated shape, while the color is considered unknown. This union is referenced by the single robot's private belief in the belief model (Fig. 18, step 1).

The tutor's utterance 'The elongated object is yellow.' is processed by the Communication SA, resulting in a new belief attributed to the tutor. This belief restricts the shape to elongated and asserts the color to be yellow. Before the belief is actually added to the belief model, the binder translates it to a binding proxy (phantom proxy) with the shape restriction as a binding feature. In the most probable configuration, the phantom proxy is bound to the existing union, which already includes the visual proxy representing the tea box (Fig. 18, step 2). The union is promptly referenced by the attributed belief and the phantom proxy is deleted soon after.

In Visual SA, the mediator intercepts the event of adding the attributed belief. The color assertion and the absence of the color restriction in the robot's belief is deemed as a

learning opportunity (the mediator knows that both beliefs reference the same binding union, hence the same object). The mediator translates the asserted color information to equivalent modal color label and compiles a learning task. The learner-recognizer uses the label and the lower level visual features of the tea box to update its yellow color model. After the learning task is complete, the mediator verifies the attributed belief, which changes its epistemic status to shared (Fig. 18, step 3). The learning action re-triggers the recognition. If the updated yellow color model is good enough, the color information in the binder and belief model is updated (Fig. 18, step 4).

A similar process also takes place in tutor assisted learning, when the robot initiates, based on an unreliable recognition, the learning process, e.g., by asking "*R: Is this red?*". In this case, the need for assistance reflects in a robot's private belief that contains the assertion about the red color and references the union representing the object. Based on this belief the Communication SA synthesizes the above question. When the robot receives the positive answer, he updates the representation of red, using a very similar mechanism as in the case of tutor driven learning.

C. Experimental results

The system was primarily developed to work in an interaction with a user. However, to comprehensively analyse the proposed learning strategies, such interactive work is time consuming and impractical. Therefore, we instead performed quantitative evaluation in simulation. The simulation environment uses stored images, which were previously captured and automatically segmented. We used a number of everyday objects, similar to those presented in Fig. 2. Each image, containing a detected and segmented object, was then manually labeled. In the learning process the tutor is replaced by an omniscient oracle, which has the ground truth data available. In this way the extensive tests could be automatically performed and a reliable evaluation of the proposed methods were obtained.

Six visual attributes were considered; four colours (red, green, blue, yellow) and two shapes (elongated, compact). The database that we used for learning contains 500 images. 400 images were used to incrementally learn the representations of six visual properties, while the rest 100 of them were used as test images. We repeated the experiment for 100 runs by randomly splitting the set of images into the training and test set and averaged the results across all runs.

During the experiment, we kept incrementally updating the representations with the training images using the Tutor driven (denoted as TD) and the Tutor assisted (denoted as TA) learning strategies. Note that in both cases the first 15 images were added in a tutor driven mode to form the initial models. At each step, we evaluated the current knowledge by recognising the visual properties of all test images. The learning performance was evaluated using two performance measures: *recognition score*, which rewards successful recognition (true positives and true negatives) and penalises incorrectly recognised visual properties (false positives and false negatives), and *tutoring costs*, which measure the level of the tutor's involvement, as defined in [26].

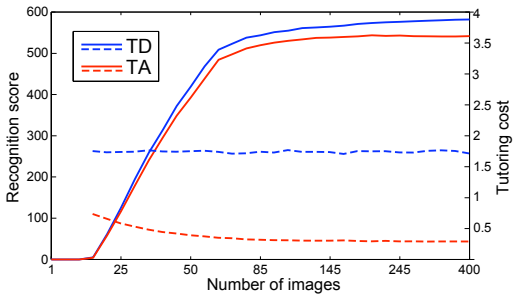


Fig. 19. Experimental results. TD: Tutor driven learning, TA: Tutor assisted learning, solid line: Recognition score, dashed line: Tutoring cost.

Fig. 19 shows the evolution of the learning performance over time for both learning strategies. The first thing to note is that the overall results improve through time. The growth of the recognition score is very rapid at the beginning when new models of newly introduced concepts are being added, and still remains positive even after all models are formed due to refinement of the corresponding representations.

Tutor-driven approach performs better, since the correct information is always given by the tutor. The inherent problem of any continuous learning framework, which involves autonomous updating of the knowledge, is propagation of errors. This is also reflected in the lower performance of the Tutor assisted approach. However, we also have to take into account the tutoring costs that occur during the learning. In Tutor-driven learning mode they are almost constant; the tutor always gives all the information about the current object, which is available. The costs of Tutor-assisted learning are significantly lower. The robot keeps asking the tutor only at the beginning of the learning process; after its knowledge gets improved the number of questions drops and most of the costs relate to the fact that the tutor has to listen to the robot and await for its questions. There is, as expected, a trade off between the quality of the results and cognitive load the tutor has to invest in the learning process. The best option would therefore be to first invoke the tutor driven approach and later on, when the models are reliable enough, switch to the tutor assisted mode.

VIII. Conclusion

In this paper we have presented a way of thinking about autonomous learning that is focussed on architectures and representations. The representational component of our theory is two-fold: on the one hand we employ representations of uncertainty and gaps in different modalities; on the other we represent how that lack of knowledge may change under action. The architectural theory is coupled: representations are shared within working memories and linked across them, again in a way that explicitly represents the different ways they might be linked. In other words our systems reason explicitly about the multiple and uncertain ways that information from different modalities might be related. We also represent novelty in the structural rules that represent the universal relationships across modalities. Finally the architectural part of the theory also describes a way that possible learning goals can be quickly ranked, so that as systems are scaled that only a feasibly small subset are actually planned for.

We have shown that this approach works, by implementing two robot systems. Each illustrates different aspects of our approach. Dora illustrates the architecture, representations of gaps and uncertainty in spatial representations, the goal management system, and the use of planning with epistemic goals. George illustrates how we explicitly represent uncertainty in multi-modal representations of a specific situation, and uncertainty and novelty in the long term model of how different modalities are related.

What are the open research issues? First of all our approach to novelty is limited. The work on KDE models provides a particular approach to this, in a particular domain, but it is far from complete. There is also a question about how constrained the tacit design knowledge makes the self-extension. At the moment Dora, and to a lesser extent George extend their models within knowledge spaces that are quite well defined. The Dora design tacitly assumes that placeholders will become places, and George has the visual features necessary to learn the correct associations with words describing colour and shape. In addition the typology we have described for different types of incompleteness is only a beginning. Most challengingly, however, we have not yet dealt with the representation of different kinds of outcome or causal incompleteness. It is in general very difficult to model and reason about these in worlds with noisy observations and noisy actions. This is because an unexpected outcome could be due to observation noise, action noise, or true novelty. Variations on latent variable models such as factored POMDPs provide a probabilistic approach, but these are notoriously difficult to learn and reason with. To identify hidden causes in models of actions is also difficult. Suppose an action of a robot fails, such as a grasping action? This could be because of picking a poor grasp position, failing to grip strongly enough, or estimating wrongly where the obejct was. These possible causes can be distinguished if the robot has the a priori notion that they are possible causes of grasp failure, but in general we want the robot to be able to discover for itself that they are possible causes. This degree of open-endedness will take many years to tackle.

In summary if an approach to self-extension based on self-understanding is to be promising as a long term approach, then we need to find ways of representing and reasoning about much more difficult knowledge gaps. We believe we have developed the first part of such an approach, and that these are indeed challenging, but achieveable goals.

ACKNOWLEDGMENT

The authors gratefully acknowledge support of the EC FP7 IST project CogX-215181.

REFERENCES

[1] N. Hawes and J. Wyatt, "Engineering intelligent information-processing systems with cast," *Advanced Engineering Informatics*, vol. 24, pp. 27–39, 2010.

[2] D. Roy, "Semiotic schemas: A framework for grounding language in action and perception," *Artificial Intelligence*, vol. 167, no. 1-2, pp. 170–205, 2005.

[3] R. Engel and N. Pfleger, "Modality fusion," in *SmartKom: Foundations of Multimodal Dialogue Systems*, W. Wahlster, Ed. Berlin: Springer, 2006, pp. 223–235.

[4] H. Jacobsson, N. Hawes, G.-J. Kruijff, and J. Wyatt, "Crossmodal content binding in information-processing architectures," in *Proc. of the 3rd International Conference on Human-Robot Interaction (HRI)*, 2008.

[5] J. Kelleher, "Integrating visual and linguistic salience for reference resolution," in *Proceedings of the 16th Irish conference on Artificial Intelligence and Cognitive Science (AICS-05)*, N. Creaney, Ed., 2005.

[6] E. Punskaya, "Bayesian approaches to multi-sensor data fusion," Master's thesis, Cambridge University Engineering Department, 1999.

[7] A. Pronobis, K. Sjöö, A. Aydemir, A. N. Bishop, and P. Jensfelt, "Representing spatial knowledge in mobile cognitive systems," Kungliga Tekniska Högskolan, CVAP/CAS, Tech. Rep. TRITA-CSC-CV 2010:1 CVAP 316, March 2010.

[8] J. Folkesson, P. Jensfelt, and H. Christensen, "The m-space feature representation for slam," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 1024–1035, Oct. 2007.

[9] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," in *4th International Symposium on Robotics Research*, 1987.

[10] A. Aydemir, A. Bishop, and P. Jensfelt, "Simultaneous object class and pose estimation for mobile robotic applications with minimalistic recognition," in *Proc. of the International Conference on Robotics and Automation (ICRA'09)*, 2010.

[11] González-Banos and Laser, "A randomized art-gallery algorithm for sensor placement," in *Proceedings of the seventeenth annual symposium on Computational geometry*, 2001.

[12] J. Castellanos, R. Martinez-Cantin, J. Tardos, and J. Neira, "Robocentric map joining: Improving the consistency of EKF-SLAM," *Robotics and Autonomous Systems*, vol. 55, no. 1, January 2007.

[13] A. N. Bishop and P. Jensfelt, "A stochastically stable solution to the problem of robocentric mapping," in *Proc. of ICRA'09*.

[14] ——, "Stochastically convergent localization of objects and actively controllable sensor-object pose," in *Proc. of the 10th European Control Conference (ECC'09)*.

[15] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. accepted, 2009.

[16] A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *The International Journal of Robotics Research (IJRR)*, vol. 29, no. 2-3, pp. 298–320, February 2010.

[17] H. Zender, O. M. Mozos, P. Jensfelt, G.-J. M. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 493–502, June 2008.

[18] C. Bäckström and B. Nebel, "Complexity results for SAS^+ planning," *Computiational Intelligence*, vol. 11, no. 4, pp. 625–655, 1995. [Online]. Available: ftp://ftp.informatik.uni-freiburg.de/papers/ki/backstrom-nebel-ci-95.ps.gz

[19] M. Brenner and B. Nebel, "Continual planning and acting in dynamic multiagent environments," *Journal of Autonomous Agents and Multiagent Systems*, vol. 19, no. 3, pp. 297–331, 2009.

[20] M. Kristan and A. Leonardis, "Multivariate online kernel density estimation," in *Computer Vision Winter Workshop*, 2010, pp. 77–86.

[21] M. Kristan, D. Skočaj, and A. Leonardis, "Online kernel density estimation for interactive learning," *Image and Vision Computing*, 2009.

[22] D. Skočaj, M. Kristan, and A. Leonardis, "Continuous learning of simple visual concepts using Incremental Kernel Density Estimation," in *VISSAP 2008*, 2008, pp. 598–604.

[23] L. P. Beaudoin and A. Sloman, "A study of motive processing and attention," in *Prospects for Artificial Intelligence: Proc. of AISB-93*, A. Sloman, D. Hogg, G. Humphreys, A. Ramsay, and D. Partridge, Eds. Amsterdam: IOS Press, 1993, pp. 229–238.

[24] M. Ozuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[25] M. Hanheide, N. Hawes, J. Wyatt, M. Göbelbecker, M. Brenner, K. Sjöö, A. Aydemir, P. Jensfelt, H. Zender, and G.-J. Kruijff, "A framework for goal generation and management," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010, submitted.

[26] D. Skočaj, M. Kristan, and A. Leonardis, "Formalization of different learning strategies in a continuous learning framework," in *EPIROB'09*, 2009, pp. 153–160.

Belief Modelling for Situation Awareness in Human-Robot Interaction [1]

Pierre Lison, Carsten Ehrler and Geert-Jan M. Kruijff

Language Technology Lab,
German Research Centre for Artificial Intelligence (DFKI GmbH)
Saarbrücken, Germany

Abstract

To interact naturally with humans, robots needs to be aware of their own surroundings. This awareness is usually encoded in some implicit or explicit representation of the situated context. In this research report, we present a new framework for constructing rich belief models of the robot's environment.

Key to our approach is the use of *Markov Logic* as a unified representation formalism. Markov Logic is a combination of first-order logic and probabilistic graphical models. Its expressive power allows us to capture both the rich relational structure of the environment and the uncertainty arising from the noise and incompleteness of low-level sensory data. Beliefs evolve dynamically over time, and are constructed by a three-fold iterative process of information fusion, refinement and abstraction. This process is reflected in distinct ontological categories. Links across these categories define the construction history by relating a belief to its ancestors. Beliefs are thus organised in a complex two-dimensional structure, with horizontal relations between belief dependents and vertical relations between belief relatives.

Beliefs also incorporate various contextual information such as spatio-temporal framing, multi-agent epistemic status, and saliency measures. Such rich annotation scheme allows us to easily interface beliefs with high-level cognitive functions such as action planning or communication. Beliefs can therefore be easily referenced, controlled and extended "top-down" by external processes to reach beyond the current perceptual horizon and include past, future or hypothetical knowledge.

Email addresses: plison@dfki.de (Pierre Lison), carsten.ehrler@dfki.de (Carsten Ehrler), gj@dfki.de (Geert-Jan M. Kruijff).

[1] This report is an extended version of a paper currently in submission to RO-MAN 2010 (19th IEEE International Symposium on Robot and Human Interactive Communication), under the same title, by the same authors.

1 Introduction

The situated context plays a central role in human-robot interaction (HRI). To be able to interact naturally with humans, robots needs to be aware of their own environment. This situation awareness is generally expressed in some sort of *belief models* in which various aspects of the external reality are encoded. Such belief models provide an explicit or implicit representation for the current state of the world, from the robot's viewpoint. They therefore serve as a representational backbone for a wide range of high-level cognitive capabilities related to reasoning, planning and learning in complex and dynamic environments. In particular, they can be used as a knowledge base by the robot to verbalise its own knowledge. Such ability is crucial to establish *transparency* in situated dialogue between the robot and one or more human interlocutor(s), for instance in socially guided learning tasks [32,30,26].

In speech-based HRI, critical tasks in dialogue understanding, management and production are directly dependent on such belief models to prime or guide their internal processing operations. Examples are context-sensitive speech recognition [20], reference resolution and generation in small- [16] and large-scale space [36], parsing of spoken dialogue [19], pragmatic interpretation [31], action selection in dialogue management [35], user-tailored response generation [34], and contextually appropriate intonation patterns in speech synthesis [18]. Contextual knowledge is also a prerequisite for the dynamic adaptation of the robot's behaviour to different environments and interlocutors [4].

Belief models are usually expressed as high level symbolic representations merging and abstracting information over multiple modalities. For human-robot interaction, the incorporated knowledge might include (inter alia):

- description of physical entities in the visual scene (what is around me);
- small- and large-scale organisation of space (what is where, where am I);
- user models (intentional and attentional state of other agents, attributed knowledge, personal profile, preferences);
- structured history of the interaction (what was said before);
- and task models (what is to be done, which actions are available).

The construction of such belief models raises two important issues for the system developer. The first question to address is how these high-level representations can be reliably abstracted from low-level sensory data [1,27]. To be meaningful, most symbolic representations must be *grounded* in (subsymbolic) sensory inputs [28]. This is a difficult problem, partly because of the noise and uncertainty contained in sensory data (partial observability), and partly because the connection between low-level perception and high-level symbols is typically difficult to formalise in a general way [8].

The second issue relates to how information arising from different modalities and time points can be efficiently *merged* into unified multi-modal structures [17], and how these inputs can refine and constrain each other to yield improved estimations, over time. This is the well-known engineering problem of multi-target, multi-sensor data fusion [7].

Belief models are thus the final product of a three-fold iterative process of information *fusion*, *refinement* and *abstraction* defined over multiple modalities and time spans. Information *fusion* refers to the operation of merging data from distinct knowledge sources into one single representation. Following the fusion operation, beliefs are then gradually *refined* – new, improved estimations are derived for each belief feature, given the collection of knowledge sources which have been merged. And finally, in complement to information refinement, beliefs are also *abstracted* by constructing high-level, amodal symbolic representations from low-level perceptual (i.e. modal) data.

1.1 Requirements for belief models in HRI

Typical HRI environments are challenging to model explicitly, as they bear the characteristics of being simultaneously *complex*, *stimuli-rich*, *multi-agent*, *dynamic* and *uncertain*. These five characteristics impose particular requirements on the nature and expressivity of the belief representations we wish to construct. Five central requirements can be formulated:

(1) HRI environments are characteristically complex, and their observation reveals a large amount of internal *structure* (for instance, spatial relations between physical entities, or possible groupings of objects according to specific properties). As a consequence, the formal representations used to specify belief models must possess the expressive power to reflect this rich relational structure in a general way, and reason over it.
(2) Physical environments are not only complex, but are also overloaded with perceptual stimuli. The robotic agent is constantly bombarbed by data coming from its sensors. Left unfiltered, the quantity of sensory information to process is sure to exhaust its computational resources. The robot must therefore be capable of actively *focusing* on the important, relevant areas while ignoring the rest. The cognitive process underlying this ability is called the *attention* system. Its role is to sort the foregrounded information from the background "clutter", on the basis of (multi-modal) saliency measures. The belief models must therefore incorporate mechanisms for computing and adapting these saliency measures over time.
(3) By definition, interactive robots are made for multi-agent settings. Making sense of communicative acts between agents requires the ability to distinguish between one's own knowledge (what I believe), knowledge at-

tributed to others (what I think the others believe), and shared common ground knowledge (what we believe as a group). Such epistemic distinctions need to be explicitly encoded in the belief representations.

(4) Situated interactions are always *dynamic* and evolve over time. This includes both the evolution of the physical environment, and the evolution of the interaction itself. The incorporation of spatio-temporal framing is thus necessary to express when and where a particular belief is supposed to hold. Spatio-temporal framing also allows us to go beyond the perceptual horizon of the "here-and-now", and link the present situation with (episodic) memories of the past, anticipation of future expected events, and hypothetical knowledge – including knowledge about distant places currently outside the reach of the robot's sensors.

(5) And last but not least, due to the partial observability of the environment (due to e.g. noise, biased measurements, occlusions), it is crucial that belief models incorporate an explicit account of *uncertainties* in order to incorporate various levels of confidence in the observed measures.

Orthogonal to these "representational" requirements, crucial performance requirements must also be adressed. To keep up with a continuously changing environment, all operations performed on the belief models (content updates, queries, etc.) must be computable under soft real-time constraints. Given the problem complexity we just outlined, this rules out the possibility of performing exact inference. An alternative, more appropriate solution is the use of *anytime* algorithms combined with various approximation methods for probabilistic inference. This constitutes our sixth and final requirement.

1.2 Gist of the approach

This report presents ongoing work on a new approach to multi-modal situation awareness which attempts to address these requirements. Key to our approach is the use of a first-order probabilistic language, *Markov Logic* [24], as a unified representation formalism to construct rich, multi-modal models of context. Markov Logic is a combination of first-order logic and probabilistic modelling. As such, it provides an elegant account of both the uncertainty and complexity of situated human-robot interactions. Our approach departs from previous work such as [13] or [27] by introducing a much richer modelling of multi-modal beliefs. Multivariate probability distributions over possible values are used to account for the partial observability of the data, while the first-order expressivity of Markov Logic allows us to consisely describe and reason over complex relational structures. As we shall see, these relational structures are annotated with various contextual information such as spatio-temporal framing (where and when is the belief valid), epistemic status (for which agents does this belief hold), and saliency (how prominent is the entity relative to

others). Furthermore, performance requirements can be addressed with approximation algorithms for probabilistic inference optimised for Markov Logic [24,23]. Such algorithms are crucial to provide an upper bound on the system latency and thus preserve its efficiency and tractability.

The rest of this report is structured as follows. Section 2 provides a brief introduction to Markov Logic, the framework used for belief modelling. Once the theoretical foundations of our work is laid out, we describe the approach itself in the sections 3 to 7. Section 3 starts by describing the software architecture in which our approach is being integrated. Section 4 details the representations which have been used to formalise the concept of "belief". Section 5 then explains step-by-step how such beliefs can be constructed bottom-up, iteratively, from perceptual inputs. Section 6 provides additional details on the attention and filtering systems. Section 7 connects beliefs to language, by showing how beliefs can be linguistically referenced, and how interaction can be used to extend beliefs with new information. Finally, Section 8 concludes this report, and provides directions for future work.

2 Markov Logic Networks

Markov logic combines first-order logic and probabilistic graphical models in a unified representation [24]. From a syntactic point of view, a *Markov logic network* L is simply defined as a set of pairs (F_i, w_i), where F_i is a first-order formula and $w_i \in \mathbb{R}$ is the associated weight of that formula.

A Markov logic network can be interpreted as a *template* for constructing Markov networks. The structure and parameters of the constructed network will vary depending on the set of constants provided to ground the predicates of the Markov Logic formulae. Such Markov network represents a probability distribution over possible words. As such, it can be used to perform probabilistic inference over the relational structure defined by the formulas F_i.

In the following, we briefly review the definition of Markov networks, and then show how they can be generated from a Markov logic network L.

2.1 Markov Network

A Markov network G, also known as a *Markov random field*, is an undirected graphical model [15] for the joint probability distribution of a set of random variables $X = (X_1, \ldots, X_n) \in \mathcal{X}$. The network G contains a node for each random variable X_i. The nodes in the network can be grouped in a set of

cliques. In graph theory, a clique is a fully connected subgraph – that is, a subset of nodes where each node is connected with each other. The joint probability distribution of the Markov network can then be factorised over the cliques of G:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \tag{1}$$

where $\phi_k(x_{\{k\}})$ is a *potential function* mapping the state of a clique k to a non-negative real value. Z is a normalization constant, known as *partition function*, and is defined as $Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$.

Alternatively, the potential function ϕ_k in (1) can be replaced by an exponentiated weighted sum over real-valued feature functions f_j:

$$P(X = x) = \frac{1}{Z} e^{\left(\sum_j w_j f_j(x)\right)} \tag{2}$$

The representation in (2) is called a *log-linear model*.

2.2 Constructing a Markov Network from a Markov Logic Network

Recall that a Markov logic network L is a set of pairs (F_i, w_i). If in addition to L we also specify a set of constants $C = \{c_1, c_2, ..., c_{|C|}\}$, one can generate a *ground Markov network* $M_{L,C}$ as follows [24]:

(1) For each possible predicate grounding over the set C, there is a binary node in $M_{L,C}$. The value of the node is true iff the ground predicate is true.
(2) For every formula F_i, there is a feature f_j for each possible grounding of F_i over C. The value of the feature $f_i(x)$ is 1 if F_i is true given x and 0 otherwise. The weight of the feature corresponds to the weight w_i associated with F_i.

The graphical representation of $M_{L,C}$ contains a node for each ground predicate. Furthermore, each formula F_i defines a set of cliques j with feature f_j over the set of distinct predicates occurring in F_i.

Following (1) and (2), the joint probability distribution of a ground Markov network $M_{L,C}$ is then given by:

$$P(X = x) = \frac{1}{Z} \prod_i \phi_i(x_{\{k\}})^{n_i(x)} = \frac{1}{Z} e^{\left(\sum_i w_i n_i(x)\right)} \tag{3}$$

The function $n_i(x)$ in (3) counts the number of true groundings of the formula F_i in $M_{L,C}$ given x.

Consider a simple Markov Logic network L made of three unary predicates, `Professor(x)`, `Teaches(x)`, and `Undergrad(x)`, and two formulae:

$$w_1 \quad \texttt{Professor(x)} \rightarrow \texttt{Teaches(x)} \tag{4}$$
$$w_2 \quad \texttt{Undergrad(x)} \rightarrow \neg\texttt{Teaches(x)} \tag{5}$$

The formulae encode the fact that most professors teach, while most undergraduate students don't. Since these two rules admit a few exceptions (professors can be on sabbatical, and some undergraduates can teach as assistants), they are specified as soft constraints with finite weights w_1 and w_2.

Assuming a particular person `A`, we can construct a ground Markov network $M_{L,\{A\}}$ over this single constant following the procedure we just outlined. The resulting network is illustrated in the Figure 1. The network $M_{L,\{A\}}$ defines a probability distribution over a set of 2^3 possible worlds (since we have three unary predicates which can be true or false, and one constant).

The probability of the world $x = (\texttt{Professor(A)}, \neg\texttt{Teaches(A)}, \neg\texttt{Undergrad(A)})$ can then be directly computed using (3). The ground Markov Network countains two features (one for each formula). In the case of world x, the first formula is violated, while the second is not. This means that $n_1(x) = 0$ and $n_2(x) = 1$. This gives us the probability $P(X = x) = \frac{1}{Z}e^{(w_1 \times 0 + w_2 \times 1)} = \frac{1}{Z}e^{w_2}$, where the partition function $Z = 4e^{w_1+w_2} + 2e^{w_1} + 2e^{w_2}$. Notice that the partition function Z grows exponentially with the weights, and will tend to infinity for large values of w_1 or w_2. If we increase the value of w_1 while keeping the value of w_2 constant, the probability $P(X = x)$ will thus approach 0.

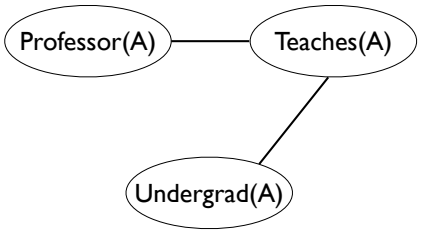


Fig. 1. Example of ground Markov Network $M_{L,C}$ given the Markov logic network $L = \langle(\texttt{Professor(x)} \rightarrow \texttt{Teaches(x)}, w_1), (\texttt{Undergrad(x)} \rightarrow \neg\texttt{Teaches(x)}, w_2)\rangle$ and the constants $C = \{A\}$. An edge between two nodes signifies that the corresponding ground atoms appear together in at least one grounding of one formula in L.

2.4 Inference

Once a Markov network $M_{L,C}$ is constructed, it can be exploited to perform conditional or MPE inference over the relational structure defined by L. A Markov Logic Network can be used to answer arbitrary queries such as "What is the probability that formula F_1 holds given that formula F_2 does?" Such query can be translated as:

$$P(F_1|F_2, L, C) = P(F_1|F_2, M_{L,C}) \tag{6}$$

$$= \frac{P(F_1 \wedge F_2|M_{L,C})}{P(F_2|M_{L,C})} \tag{7}$$

$$= \frac{\sum_{x \in \mathcal{X}_{P_1} \cap \mathcal{X}_{P_2}} P(X = x|M_{L,C})}{\sum_{x \in \mathcal{X}_{P_2}} P(X = x|M_{L,C})} \tag{8}$$

where $\mathcal{X}_{P_i}$ represent the set of worlds where the formula F_i holds.

Exact inference in Markov Networks is a #P-complete problem [15] and is thus untractable. However, several efficient algorithms for probabilistic inference such as weighted MAX-SAT, Markov Chain Monte Carlo (MCMC) or lifted belief propagation can then be used to yield approximate solutions [23,25,29]. Given the requirements of our application domain (see Section 1), and particularly the need to operate under soft real-time constraints, such approximation methods are an absolute necessity.

2.5 Learning

The weight w_i in a Markov logic network encode the "strength" of its associated formula F_i. In the limiting case, where $\lim_{w_i \to \infty}$, the probability of a world violating F_i has zero probability. For smaller values of the weight, worlds violating the formula will have a low, but non-zero probability.

But how are these weights specified? In most cases, weights are learned based on training samples extracted from a relational database. Several machine learning algorithms for parameter learning can be applied to this end, from classical gradient-based techniques to more sophisticated algorithms specifically designed for statistical relational learning [21,12].

In addition to weights learning, it is also possible to learn the *structure* of a Markov Logic problem, either partially (by adding additional clauses to the network or refining the existing ones), or completely (by learning a full

network from scratch). Structure learning is usually performed with algorithms borrowed from Inductive Logic Programming [14,22].

3 Architecture

3.1 Global schema

Our approach is being developed as part of a *distributed cognitive architecture* for autonomous robots in open-ended environments [9].

The architectural schema is based on a distributed set of subarchitectures. Each subarchitecture encapsulates a number of processing components running in parallel. The components can access sensors, effectors, as well as a blackboard (working memory) available for the whole subarchitecture. Via this central working memory, each component is able to asynchronously read and update shared information within the subarchitecture. The information flow between components is thus based on the idea of *parallel refinement of shared representations*, eschewing the standard point-to-point connectivity of traditional message-based frameworks.

The components can be either unmanaged (data-driven) or managed (goal-driven). Goal-driven components can be controlled explicitly at runtime by a task manager specifying which component is allowed to run at a given time. This explicit control of information and processing is crucial to dynamically balance and constrain the computational load among components.

Finally, subarchitectures can also communicate with each other by accessing (reading, inserting, updating) their respective working memories.

3.2 Implementation of the schema

This architectural schema has been fully implemented in a software toolkit called **CAST** [9], which has been developed to support the construction and exploration of information-processing architectures for intelligent systems such as robots. Components can be implemented in Java, C++, or Python, while the shared data structures in the working memory are specified in a language-neutral specification using ICE [2] [11].

[2] Internet Communications Engine, an object-oriented middleware developed by ZeroC: `http://www.zeroc.com/ice.html`

The architecture and its associated toolkit have been applied to various scenarios such as visual learning and object manipulation in a tabletop scene [33] and exploration of indoor environments for human-augmented mapping [10].

3.3 Binder subarchitecture

Our approach to multi-modal belief modelling is implemented in a specific subarchitecture developed under the CAST framework. This subarchitecture is called the "*binder*". The binder is directly connected to the other subarchitectures (i.e. vision, navigation, manipulation, etc.), and serves as a central hub for the information gathered about the environment. The core of the binder is its working memory, where beliefs are formed from incoming perceptual inputs, and are then iteratively fused, refined and abstracted to yield stable, high-level beliefs.

The resulting beliefs can also be easily accessed and retrieved by the other subarchitectures. Such retrieval operation allows each subarchitecture to use the binder as a system-wide information repository about the world state. The beliefs can then be directly exploited by high-level cognitive functions such as planning, cross-modal learning or communication. They can also be used by perceptual components to adapt their internal processing operations to the current situated context (contextual priming, anticipation, etc.)

Fig. 2 schematically illustrates the interface between the binder system and the rest of the software architecture.

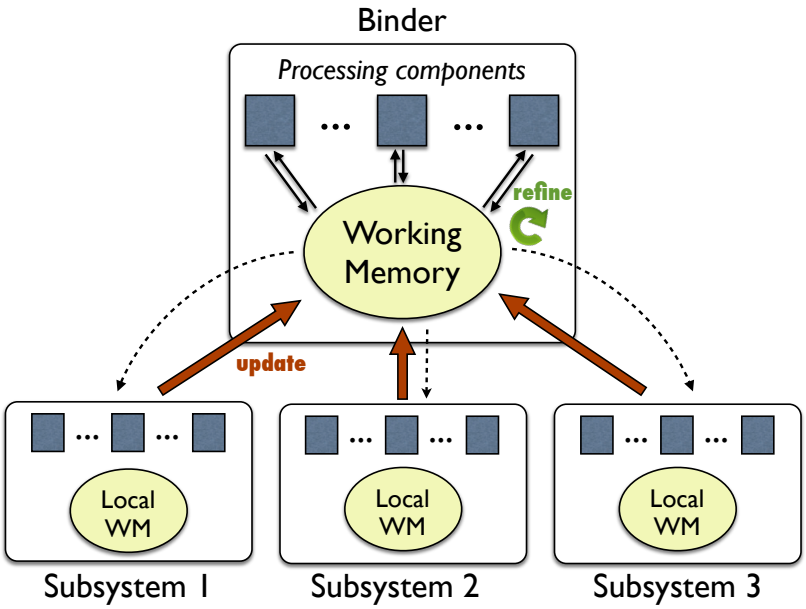


Fig. 2. Schema of the cognitive architecture in relation with the binder

The data structures included in the binder are inherently probabilistic – each feature or information bit pertaining to an entity can be associated to a probability value, reflecting the confidence level of the subsystem. This enables the system to deal with varying levels of noice and uncertainty, which are pervasive and unavoidable for most sensory-motric processes.

Now that the software architecture of our approach has been described, the next section proceeds by detailing how beliefs are represented in the binder, and how this representation is precisely formalised.

4 Representation of beliefs

Each unit of information manipulated by the binder is expressed as a *probability distribution* over a space of possible values. Such unit of information is called a **belief**.

Beliefs are constrained both *spatio-temporally* and *epistemically*. They include a frame stating where and when the information is assumed to be valid, and an epistemic status stating for which agent(s) the information holds.

Formally, a **belief** is a tuple $\langle i, e, \sigma, c, \boldsymbol{\delta}, h \rangle$, where i is the belief identifier, e is an epistemic status, σ a spatio-temporal frame, c an ontological category, $\boldsymbol{\delta}$ is the belief content (specified as a probability distribution), and h is the history of the belief.

We describe below each of these components one by one.

4.1 Epistemic status e

Interactive robots must be able to distinguish between their own knowledge, knowledge of others, and shared knowledge (common ground). We specify such information in the epistemic status of the belief. For a given agent a, the **epistemic status** e can be either:

- *private*, denoted $\mathsf{K}\{a\}$: private beliefs come from within the agent a. In other words, they are a direct or indirect result of agent a's perception of the environment;
- *attributed*, denoted $\mathsf{K}\{a[b_1, ..., b_n]\}$: Attributed beliefs are beliefs which are ascribed to other agents. They are a's conjecture about the mental states of other agents $b_1, ..., b_n$, usually as a result of a's interpretations of previous communicative acts performed by $b_1, ..., b_n$.

- *shared*, denoted $\mathsf{K}\{a_1, ..., a_m\}$: Shared beliefs contain information which is part of the common ground for the group [3].

Shared epistemic status subsumes both private and attribute epistemic status. A shared belief $\mathsf{K}\{a, b\}$ therefore also implies the two private beliefs $\mathsf{K}\{a\}$ and $\mathsf{K}\{b\}$ and the two attributed beliefs $\mathsf{K}\{a[b]\}$ and $\mathsf{K}\{b[a]\}$.

4.2 Spatio-temporal frame σ

The **spatio-temporal frame** σ defines a contiguous spatio-temporal interval, the nature of which depends on the application domain. In the simplest case, the spatial dimension can be modelled by a discrete set of regions and the temporal dimension via intervals defined on real-valued time points. Of course, more complex spatio-temporal modelling can be designed. The regions in the spatial dimension can be hierarchically organised (e.g. based on a spatial ontology) instead of being defined as flat list of possible regions. The temporal dimension can be adapted in a similar way.

Moreover, the spatio-temporal frame can be extended with the notion of *perspective*, where spatial and temporal constraints are defined as being *relative* to a particular agent a. Using the notion of perspective, we can capture the fact that each agent view the environment in its own specific way (i.e. the object which is on my left might be to the right of the robot).

It is important to note that beliefs can express past or future knowledge (i.e. memories and anticipations). That is, beliefs need not be directly grounded in the "here-and-now" observations.

4.3 Ontological category c

The **ontological category** is used to sort the various belief types which can be created. Various levels of beliefs are defined, from the lowest to the highest abstraction level. Figure 5 illustrates the role of these categories in the belief formation process.

(1) The lowest-level type of beliefs is the *percept* (or *perceptual belief*), which is a uni-modal representation of a given entity [3] or relation between entities in the environment. Perceptual beliefs are inserted onto the binder by the various subsystems included in the architecture. The epistemic

[3] The term "entity" should be understood here in a very general sense. An entity can be an object, a place, a landmark, a person, etc.

status of a percept is private per default, and the spatio-temporal frame is the robot's present place and time-point.

(2) If several percepts (from distinct modalities) are assumed to originate from the same entity, they can be grouped into a *percept union*. A percept union is just another belief, whose content is the combination of all the features from the included percepts.

(3) The features of a percept union can be abstracted using multi-modal fusion and yield a *multi-modal belief*.

(4) If the current multi-modal belief (which is constrained to the present spatio-temporal frame) is combined with beliefs encoded in past or future spatio-temporal frames, it forms a *temporal union*.

(5) Finally, the temporal unions can be refined *over time* to improve the estimations, leading to a *stable belief*, which is both multi-modal and spans an extended spatio-temporal frame.

4.4 Belief content δ

The **distribution** δ defines the possible content values for the belief. In general, each alternative value can be expressed as a (propositional) logical formula. In most practical cases, such formula can be represented as a flat list of features. The feature values can be either discrete (as for categorical knowledge) or continuous (as for real-valued measures).

A feature value can also specify a *pointer* to another belief, allowing us to capture the relational structure of the environment we want to model. The resulting relational structure can be of arbitrary complexity.

Discrete probability distributions can be expressed as a set of pairs $\langle \varphi, p \rangle$ with φ a formula, and p a probability value, where the values of p must satisfy the usual constraints for probability values. For continuous distribution, we generally assume a known distribution (for instance, a normal distribution) combined with the required parameters (e.g. its mean and variance).

In practice, maintaining a single big distribution over all possible values of the belief is both computationally expensive and unecessary. The distribution can usually be decomposed into a list of smaller distributions over parts of the belief content. This can be done by breaking down the formulae into elementary predications, and assuming conditional independence between these elementary predicates. The probability distribution $\boldsymbol{\delta}$ can then be factored into smaller distributions $\boldsymbol{\delta}_1 ... \boldsymbol{\delta}_n$.

4.5 Belief history h

Finally, via the **belief history** h, each belief contains bookkeeping information detailing the history of its formation. This is expressed as two set of pointers: one set of pointers to the belief ancestors (i.e. the beliefs which contributed to the emergence of this particular belief) and one set of pointers to the belief offspring (the ones which themselves emerged out of this particular belief).

Beliefs are thus organised in a complex two-dimensional structure, with horizontal relations between beliefs of same category (representing the relational structure of the world), and vertical relations between a belief and its parents/offpsring (representing the historical evolution of a given belief as they are processed by the system).

Perceptual beliefs have by construction no belief parent. Instead, they include in their belief history a pointer to the local data structure in the subarchitecture which was at the origin of the belief.

4.6 Example of belief representation

Consider an environment with a blue mug such as the one pictured in Figure 3. The mug is perceived by the robot sensors (for instance, by one binocular camera, or by a haptic sensor mounted on a robotic arm). Sensory data is extracted and processed by the sensory subarchitecture(s). A the end of the process, a perceptual belief is created, with four features: object label, colour, location, and height.

Fig. 3. A blue mug

Due to the noise and uncertainty of sensory data, the perceived characteristics of the object are uncertain. Let us assume for our example two uncertainties:

- The colour value of the object is uncertain (the vision system hesitates between blue with probability 0.77 and purple with probability 0.22),
- and the recognition of the object itself is also uncertain (the recognised object might be a false positive with no corresponding entity in the real world. The probability of a false positive is 0.1).

Such perceptual belief i would be formally defined as:

$$\langle i, \{\text{robot}\}, \sigma_{[\mathsf{here\text{-}and\text{-}now}]}, \mathsf{percept}, \boldsymbol{\delta}, h \rangle \tag{9}$$

with a probability distribution $\boldsymbol{\delta}$ containing three alternative formulae φ_1, φ_2 and φ_3. A graphical illustration of the belief i is provided in Figure 4.

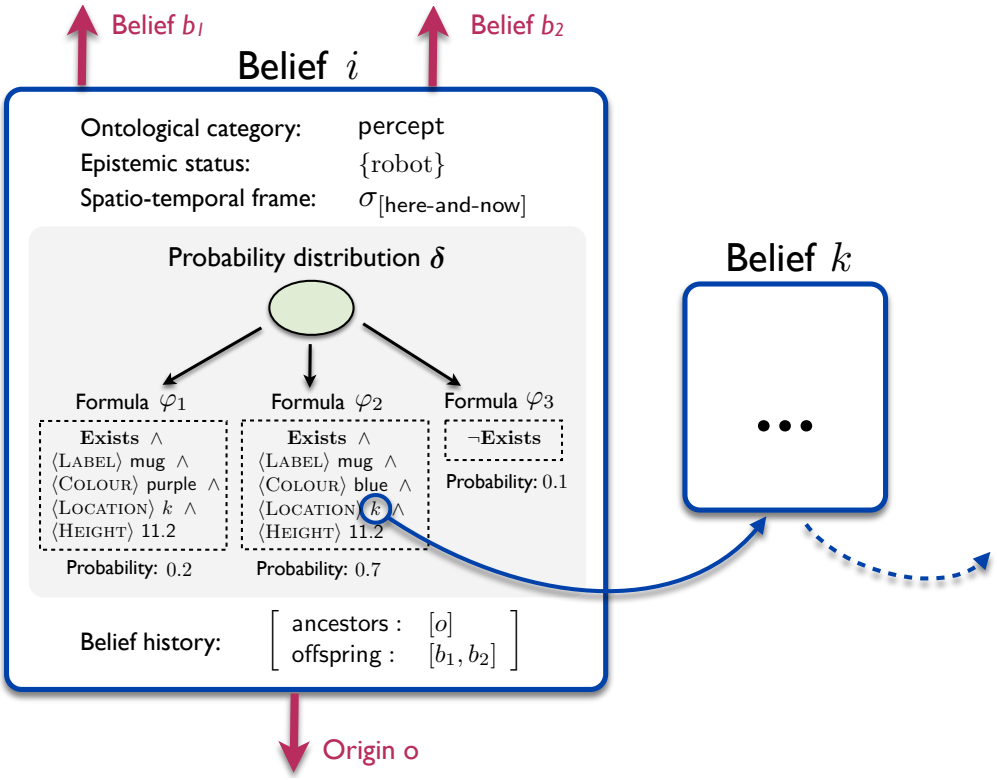


Fig. 4. Schematic view of a belief representation.

We can see in Figure 4 that the formula φ_2 specifies the existence (with probability 0.7) of a blue mug entity of size 11.2 cm, at location k, perceived by the robot in the current spatio-temporal frame ("here-and-now"). Notice that the location is described as a pointer to another belief k. Such pointers are crucial to capture relational structures between entities.

The belief i also specifies a belief history h. The belief i being a percept, its history is defined as a pointer to a local data structure o in the subarchitecture responsible for the belief's creation. The belief history also contains two pointers b_1 and b_2 to the belief's offspring.

4.7 Alternative formalisation

The logically inclined reader might notice that the belief representation we outlined can also be equivalently formalised with a hybrid logic [2] complemented by a probability language [5]. The belief $\langle i, e, \sigma, c, \boldsymbol{\delta}, h\rangle$ is then expressed as:

$$\mathsf{K}e/\sigma : \bigwedge_{\langle\varphi,p\rangle\in\boldsymbol{\delta}} (P(@_{\{i:c\}}\varphi) = p) \ \wedge \ \exists!\ \langle\varphi, p\rangle \in \boldsymbol{\delta} : (@_{\{i:c\}}\varphi) \tag{10}$$

where @ is the satisfiability operator from hybrid logic. The advantage of using such representation is the possibility of using logical inference mechanisms to *reason* over such structure and automatically derive new beliefs. We leave this question as an interesting area of future research.

4.8 Conversion into Markov Logic

The conversion of the probability distribution $\boldsymbol{\delta}$ into Markov Logic formulae is relatively straightforward. Modal operators are translated into first-order predicates and nominals into contants. A (sub-)formula such as:

$$\langle \textsc{Colour} \rangle \ \mathsf{blue} \tag{11}$$

which is declared true with probability p_1 within a belief i is therefore expressed as the following Markov Logic formula:

$$w_1 \quad \texttt{Colour}(\texttt{I}, \texttt{Blue}) \tag{12}$$

where the weight $w_1 = \log \dfrac{p_1}{1 - p_1}$.

5 Bottom-up belief construction

We now turn our attention to the way a belief model can be constructed bottom-up from the initial input provided by the perceptual beliefs. The formation of belief models proceeds in four consecutive steps: (1) *perceptual grouping*, (2) *multi-modal fusion*, (3) *tracking* and (4) *temporal smoothing*. Figure 5 provides a graphical illustration of this process.

5.1 Perceptual grouping

The first step is to decide which percepts from different modalities belong to the same real-world entity, and should therefore be grouped into a belief. For a pair of two percepts p_1 and p_2, we infer the likelihood of these two percepts being generated from the same underlying entity in the real-world. This is realised by checking whether their respective features *correlate* with each other.

The probability of these correlations are encoded in a Markov Logic Network. The formulae might for instance express a high compatibility between the

haptic feature "shape: cylindrical" and the visual feature "object: mug" (since most mugs are cylindrical), but a very low compatibility between the features "shape: cylindrical" and "object: ball". Eq. (13) illustrates the correlation between the cylindrical shape (`Cyl`) and the object label "mug" (`Mug`).

$$w_i \quad \texttt{Shape}(\texttt{x}, \texttt{Cyl}) \wedge \texttt{Label}(\texttt{y}, \texttt{Mug}) \rightarrow \texttt{Unify}(\texttt{x}, \texttt{y}) \tag{13}$$

Markov Logic formulae can also express incompatibility between features, for instance between a spherical shape and a object labelled as a mug:

$$w_j \quad \texttt{Shape}(\texttt{x}, \texttt{Spherical}) \wedge \texttt{Label}(\texttt{y}, \texttt{Mug}) \rightarrow \neg\texttt{Unify}(\texttt{x}, \texttt{y}) \tag{14}$$

Additional formulae are used to specify generic requirements on the perceptual grouping process, for instance that `x` and `y` must be distinct beliefs and originate from distinct subarchitectures. The prior probability of a grouping is also specified as a Markov Logic formula.

A grouping of two percepts will be given a high probability if (1) one or more feature pairs correlate with each other, and (2) there are no incompatible feature pairs. This perceptual grouping process is triggered at each insertion or update of percepts on the binder (provided the number of modalities in the

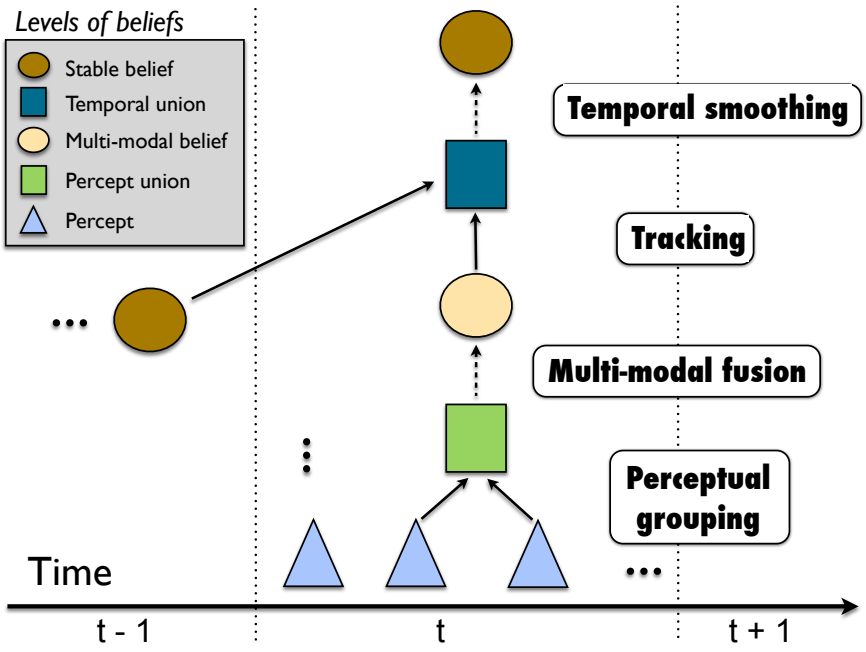


Fig. 5. Bottom-up belief model formation.

system > 1). The outcome is a set of possible unions, each of which has an existence probability describing the likelihood of the grouping.

5.2 Multi-modal fusion

We want multi-modal beliefs to go beyond the simple superposition of isolated modal contents. Multi-modal information should be *fused*. In other words, the modalities should co-constrain and refine each other, yielding new multi-modal estimations which are globally more accurate than the uni-modal ones.

Multi-modal fusion is also specified in a Markov Logic Network. As an illustration, assume a multi-modal belief `B` with a predicate $\texttt{Position}(\texttt{B}, \texttt{loc})$ expressing the positional coordinates of an entity, and assume the value `loc` can be estimated via distinct modalities a and b by way of two predicates $\texttt{Position}_{(\texttt{a})}(\texttt{U}, \texttt{loc})$ and $\texttt{Position}_{(\texttt{b})}(\texttt{U}, \texttt{loc})$ included in a percept union `U`.

$$w_i \quad \texttt{Position}_{(\texttt{a})}(\texttt{U}, \texttt{loc}) \rightarrow \texttt{Position}(\texttt{B}, \texttt{loc}) \tag{15}$$

$$w_j \quad \texttt{Position}_{(\texttt{b})}(\texttt{U}, \texttt{loc}) \rightarrow \texttt{Position}(\texttt{B}, \texttt{loc}) \tag{16}$$

The weights w_i and w_j specify the relative confidence of the measurements for the modality a and b, respectively.

5.3 Tracking

Environments are dynamic and evolve over time – and so should beliefs. Analogous to perceptual grouping which seeks to bind observations over modalities, tracking seeks to bind beliefs *over time*. Both past beliefs (memorisation) and future beliefs (anticipation) are considered. The outcome of the tracking step is a distribution over temporal unions, which are combinations of beliefs from different spatio-temporal frames.

The Markov Logic Network for tracking works as follows. First, the newly created belief is compared to the already existing beliefs for similarity. The similarity of a pair of beliefs is based on the correlation of their content (and spatial frame), plus other parameters such as the time distance between beliefs.

Eq. (17) illustrates a simple example where two beliefs are compared on their shape feature to determine their potential similarity:

$$w_i \quad \texttt{Shape}(\texttt{x}, \texttt{Cyl}) \wedge \texttt{Shape}(\texttt{y}, \texttt{Cyl}) \rightarrow \texttt{Unify}(\texttt{x}, \texttt{y}) \tag{17}$$

If two beliefs $\mathtt{B_1}$ and $\mathtt{B_2}$ turn out to be similar, they can be grouped in a temporal union $\mathtt{U}$ whose temporal interval is defined as $[\mathtt{start}(\mathtt{B_1}), \mathtt{end}(\mathtt{B_2})]$.

5.4 Temporal smoothing

Finally, temporal smoothing is used to refine the estimates of the belief content *over time*. Parameters such as recency have to be taken into account, in order to discard outdated observations.

The Markov Logic Network for temporal smoothing is similar to the one used for multi-modal fusion:

$$w_i \quad \mathtt{Position}_{(\mathtt{t\text{-}1})}(\mathtt{U}, \mathtt{loc}) \rightarrow \mathtt{Position}(\mathtt{B}, \mathtt{loc}) \tag{18}$$

$$w_j \quad \mathtt{Position}_{(\mathtt{t})}(\mathtt{U}, \mathtt{loc}) \rightarrow \mathtt{Position}(\mathtt{B}, \mathtt{loc}) \tag{19}$$

6 Attention and filtering

As we mentioned in the introduction, an *active* perception of the environment relies on the ability to focus the robot's sensing activities to the relevant entities in its surroundings, while ignoring the rest. Moreover, it is crucial for performance reasons to perform aggressive filtering on the beliefs manipulated by the binder, in order to retain only the most likely ones, and pruning the others. This section explores these two issues.

6.1 Salience modelling

The attention system is driven in our approach by *saliency* measures. These measures are represented in the binder as a specific feature included in the belief content. The salience value gives an estimate of the "prominence" or quality of standing out of a particular entity relative to neighboring ones. It allows us to guide the attentional behaviour of the agent by specifying which entities are currently in focus. The resolution of referring expressions containing deictic demonstratives such as "this" and "that" is for instance directly dependent on the salience levels of related entities.

In our model, the salience is defined as a real-valued measure which combines several perceptual measures such as the object size and its linear and angular distances relative to the robot. During linguistic interaction, these perceptual

measures can be completed by measures of linguistic saliency, such as the recency of the last reference to the object.

The salience being defined as a real-valued scalar, its probability is defined as a density function $\Re \rightarrow [0, 1]$.

6.2 Belief filtering

Techniques for *belief filtering* are essential to keep the system tractable. Given the probabilistic nature of the framework, the number of beliefs is likely to grow exponentially over time. Most of these beliefs will have a near-zero probability. A filtering system can effectively prune such unecessary beliefs, either by applying a minimal probability threshold on them, or by maintaining a fixed maximal number of beliefs in the system at a given time. Naturally, a combination of both mechanisms is also possible.

In addition to filtering techniques, *forgetting* techniques could also improve the system efficiency [6].

7 Referencing and top-down extension

7.1 Referencing beliefs

Beliefs are high-level symbolic representations available for the whole cognitive architecture. As such, they provide an unified model of the environment which can be used during interaction. An important aspect of this is *reference resolution*, which connects linguistic expressions such as "this box" or "the ball on the floor" to the corresponding beliefs about entities in the environment.

Reference resolution is performed via a Markov Logic Network specifying the correlations between the linguistic constraints of the referring expression and the belief features – in particular, the entity *saliency* and its associated *categorical* knowledge.

Eq. (20) illustrates the resolution of a referring expression R with the linguistic label "mug" to a belief B which includes a label feature with value Mug:

$$w_i \quad (\texttt{Label}(\texttt{B}, \texttt{Mug}) \ \wedge \ \texttt{RefLabel}(\texttt{R}, \texttt{Mug})) \rightarrow \ \texttt{Resolve}(\texttt{R}, \texttt{B}) \tag{20}$$

The resolution process yields a distribution over alternative referents, which is then retrieved by the communication subsystem for further interpretation.

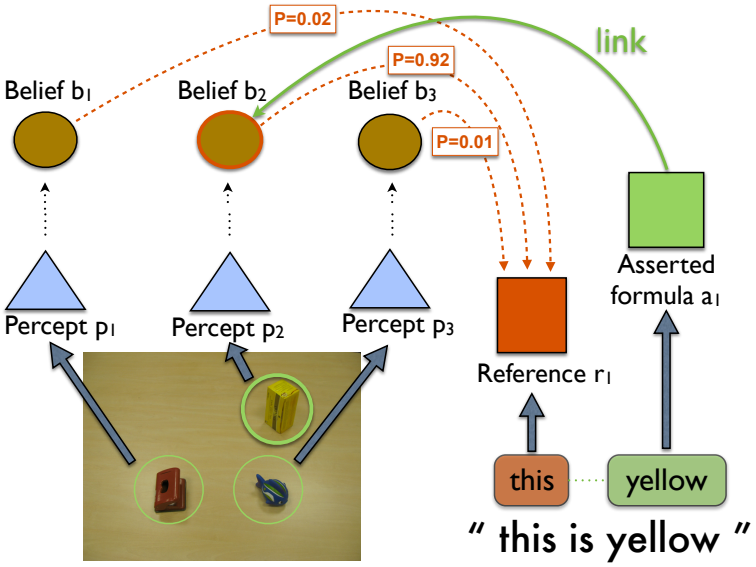


Fig. 6. An utterance such as "this is yellow" illustrates the two mechanisms of referencing and belief extension. First, the expression "this" is resolved to a particular entity. Since "this" is a deictic, the resolution is performed on basis of saliency measures. The belief $\mathtt{B}_2$ is selected as most likely referent. Second, the utterance also provides new information – namely that the object is yellow. This asserted content must be incorporated into the robot's beliefs. This is done by constructing a new belief which is linked (via a pointer) to the one of the referred-to entity.

7.2 Asserting new information

In Section 5, we described how beliefs can be formed from percepts, bottom-up. When dealing with cognitive robots able to reflect on their own experience, anticipate possible events, and communicate with humans to improve their understanding, beliefs can also be manipulated "top-down" via high-level cognitive functions such as reasoning, planning, learning and interacting.

We concentrate here on the question of belief extension via interaction. In addition to simple reference, interacting with a human user can also provide *new* content to the beliefs. Using communication, the human user can directly extend the robot's current beliefs, in a top-down manner, without altering the incoming percepts. The epistemic status of this information is *attributed*. If this new information conflicts with existing knowledge, the agent can decide to trigger a clarification request to resolve the conflict.

Fig. 6 provides an example of reference resolution coupled with a belief extension, based on the utterance "this is yellow".

8 Conclusion

In this report, we presented a new approach to the construction of *rich belief models* for situation awareness. These beliefs models are spatio-temporally framed and include epistemic information for multi-agent settings. Markov Logic is used as a unified representation formalism, allowing us to capture both the complexity (relational structure) and uncertainty (partial observability) of typical HRI application domains.

The implementation of the approach outlined in this report is ongoing. We are using the Alchemy software [4] for efficient probabilistic inference. The binder system revolves around a central working memory where percepts can be inserted, modified or deleted. The beliefs are automatically updated to reflect the incoming information. A GUI can be used to monitor and control at runtime the binder behaviour.

Besides the implementation, future work will focus on three aspects. The first aspect pertains to the use of *machine learning techniques* to learn the model parameters. Using statistical relational learning techniques and a set of training examples, it is possible to learn the weights of a given Markov Logic Network [24]. The second aspect concerns the extension of our approach to non-indexical epistemic knowledge – i.e. the representation of *events*, *intentions*, and *plans*. Finally, we want to evaluate the empirical performance and scalability of our approach under a set of controlled experiments.

Acknowledgments

The research leading to these results has received funding from the European Union's 7th Framework Programme [FP7/2007-2013] under grant agreement No. 215181 (CogX). The authors wish to thank Michael Brenner, Marc Hanheide, Jeremy Wyatt, Miroslav Janíček, Hendrik Zender and many other researchers from the CogX consortium for useful discussions.

References

[1] L. W. Barsalou. Abstraction in perceptual symbol systems. *Philosophical Transactions of the Royal Society of London: Biological Sciences*, 358:1177–1187, 2003.

[4] Cf. `http://alchemy.cs.washington.edu/`

[2] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–625, 2000.

[3] H. H. Clark and E. F. Schaefer. Contributing to discourse. *Cognitive Science*, 13:259–294, 1989.

[4] F. Doshi and N. Roy. Spoken language interaction with model uncertainty: an adaptive human-robot interaction system. *Connection Science*, 20(4):299–318, 2008.

[5] R. Fagin, Joseph Y. Halpern, and Nimrod Megiddo. A logic for reasoning about probabilities. *Inf. Comput.*, 87(1-2):78–128, 1990.

[6] S. T. Freedman and J. A. Adams. Human-inspired robotic forgetting: Filtering to improve estimation accuracy. In *Proceedings of the 14^{th} IASTED International Conference on Robotics and Applications*, pages 434–441, 2009.

[7] D. L. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, August 2002.

[8] S. Harnad. The symbol grounding problem, 1990.

[9] N. Hawes and J. Wyatt. Engineering intelligent information-processing systems with cast. *Advanced Engineering Infomatics*, To Appear.

[10] N. Hawes, H. Zender, K. Sjöö, M. Brenner, G.-J. M. Kruijff, and P. Jensfelt. Planning and acting with an integrated sense of space. In *Proceedings of the 1st International Workshop on Hybrid Control of Autonomous Systems – Integrating Learning, Deliberation and Reactive Control (HYCAS)*, pages 25–32, Pasadena, CA, USA, July 2009.

[11] Michi Henning. A new approach to object-oriented middleware. *IEEE Internet Computing*, 8(1):66–75, 2004.

[12] Tuyen N. Huynh and Raymond J. Mooney. Max-margin weight learning for markov logic networks. In *ECML PKDD '09: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 564–579, Berlin, Heidelberg, 2009. Springer-Verlag.

[13] H. Jacobsson, N.A. Hawes, G.-J. M. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. In *Proceedings of the 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Amsterdam, The Netherlands, March 12–15 2008.

[14] Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 441–448, New York, NY, USA, 2005. ACM.

[15] D. Koller, N. Friedman, L. Getoor, and B. Taskar. Graphical models in a nutshell. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[16] G-.J. M. Kruijff, J.D. Kelleher, and N. Hawes. Information fusion for visual reference resolution in dynamic situated dialogue. In *Perception and Interactive Technologies (PIT 2006)*. Spring Verlag, 2006.

[17] G.-J. M. Kruijff, John D. Kelleher, and N. Hawes. Information fusion for visual reference resolution in dynamic situated dialogue. In *Perception and Interactive*

Technologies: International Tutorial and Research Workshop, PIT 2006, volume 4021 of *Lecture Notes in Computer Science*, pages 117 – 128, Kloster Irsee, Germany, June 2006. Springer Berlin / Heidelberg.

[18] I. Kruijff-Korbayová, S. Ericsson, K. J. Rodríguez, and E. Karagjosova. Producing contextually appropriate intonation in an information-state based dialogue system. In *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, pages 227–234, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[19] P. Lison. Robust processing of situated spoken dialogue. In *Von der Form zur Bedeutung: Texte automatisch verarbeiten / From Form to Meaning: Processing Texts Automatically*. Narr Verlag, 2009. Proceedings of the GSCL Conference 2009 , Potsdam, Germany.

[20] P. Lison and G.-J. M. Kruijff. Salience-driven contextual priming of speech recognition for human-robot interaction. In *Proceedings of ECAI 2008*, Athens, Greece, 2008.

[21] Daniel Lowd and Pedro Domingos. Efficient weight learning for markov logic networks. In *PKDD 2007: Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, pages 200–211, Berlin, Heidelberg, 2007. Springer-Verlag.

[22] Lilyana Mihalkova and Raymond J. Mooney. Bottom-up learning of markov logic network structure. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 625–632, New York, NY, USA, 2007. ACM.

[23] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, pages 458–463. AAAI Press, 2006.

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

[25] Sebastian Riedel. Improving the accuracy and efficiency of MAP inference for markov logic. pages 468–475, 2008.

[26] D. Roy. Learning words and syntax for a scene description task. *Computer Speech and Language*, 16(3), 2002.

[27] D. Roy. Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1-2):170–205, 2005.

[28] D. Roy and E. Reiter. Connecting language to the world. *Artificial Intelligence*, 167(1-2):1–12, 2005.

[29] Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pages 1094–1099. AAAI Press, 2008.

[30] D. Skočaj, G. Berginc, B. Ridge, A. Štimec, M. Jogan, O. Vanek, A. Leonardis, M. Hutter, and N. Hewes. A system for continuous learning of visual concepts. In *International Conference on Computer Vision Systems ICVS 2007*, Bielefeld, Germany, March 2007.

[31] M. Stone and R.H. Thomason. Context in abductive interpretation. In *Proceedings of EDILOG 2002: 6th workshop on the semantics and pragmatics of dialogue*, 2002.

[32] Andrea L. Thomaz. *Socially Guided Machine Learning*. PhD thesis, MIT, 2006.

[33] A. Vrečko, D. Skočaj, N. Hawes, and A. Leonardis. A computer vision integration model for a multi-modal cognitive system. In *IEEE/RSJ International Conference on Intelligent RObots and Systems*, pages 3140–3147, 2009.

[34] M. Walker, S. Whittaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy. Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28(5):811–840, 2004.

[35] J. Williams and S. Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):231–422, 2007.

[36] H. Zender, G.-J. M. Kruijff, and I. Kruijff-Korbayová. Situated resolution and generation of spatial referring expressions for robotic assistants. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*, pages 1604–1609, Pasadena, CA, USA, July 2009.