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

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

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

University of Birmingham
⟨jlw@cs.bham.ac.uk⟩

Due date of deliverable:	March 31 2010
Actual submission date:	March 31 2010
Lead partner:	BHAM
Revision:	v1
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 18 months of the project, and set out a typology of epistemic knowledge. 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

Role of representations of beliefs in CogX

Contribution to the CogX scenarios and prototypes

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

1.3 Relation to the state-of-the-art

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, and the uncertainty in its knowledge. 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*. 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. We illustrate these, together with some examples for each in Figure 2.

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 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 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.

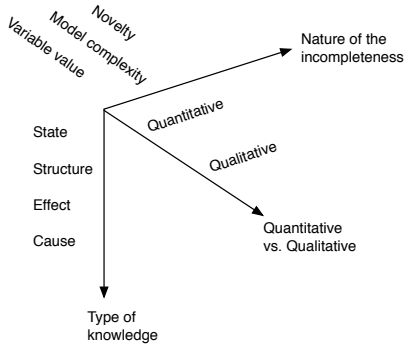


Figure 1: Three dimensions of variation in gaps and uncertainty.

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.

1. State 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 gent 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). The space here being the set of natural numbers. 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 density over some subset of the natural numbers. 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, in other words we have an experience that is extrapolating our previous range of experience.

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 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.

8. Effect model complexity: it may simply be uncertain as to how many effects of an action there are.

9. Effect novelty: it may be that an action has been taken, and that its effect or outcome is novel in some respect.

10. Causal value uncertainty: 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.

11. Causal model complexity: here it is unknown how many variables may influence the outcome of an action or event, and the precise relationship between them may be unknown.

12. Causal novelty: here is it known that an action which normally has a reliable effect has an unexpected outcome, i.e. there is a suprise, and that there must therefore be a cause of that suprising 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 belief models of the robot's environment. 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 constructred 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.

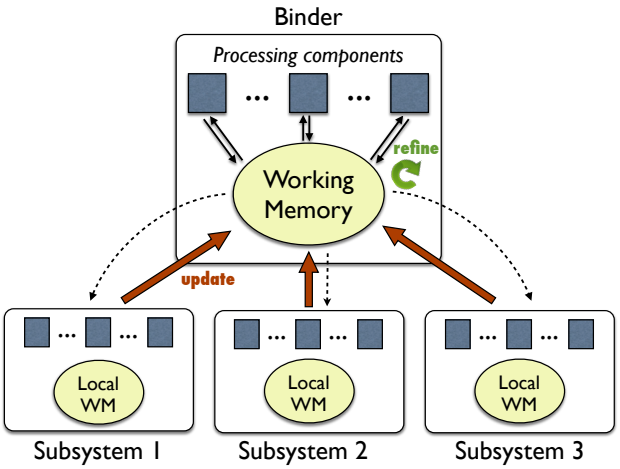


Figure 2: Schema of the cognitive architecture in relation with the binder

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 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 [8].

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*.

[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.

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 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

Figure 3: A blue mug

the end of the process, a perceptual belief is created, with four features: object label, colour, location, and height.

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.

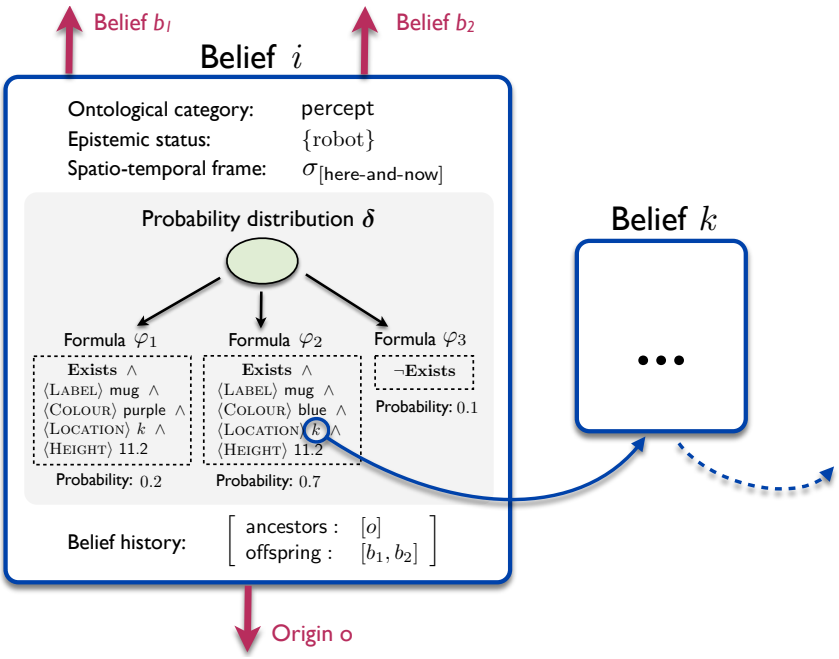


Figure 4: Schematic view of a belief representation.

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.

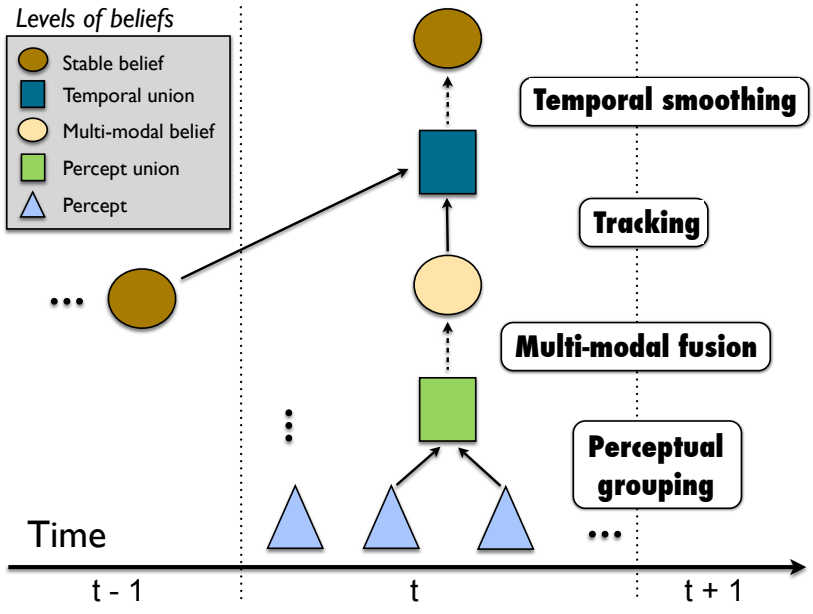


Figure 5: Bottom-up belief model formation.

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. 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`).

$$w_i \quad \texttt{Shape}(\texttt{x}, \texttt{Cyl}) \wedge \texttt{Label}(\texttt{y}, \texttt{Mug}) \rightarrow \texttt{Unify}(\texttt{x}, \texttt{y}) \tag{2}$$

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

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

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.

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 `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.

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

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

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:

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

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

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:

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

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

4 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.

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. Discretization drastically reduces the number of states that have to be considered e.g. during the planning process [13] and serves as a basis for higher level conceptualization [31].

This section explains how different domains of spatial knowledge are represented in our system. The representation focuses on structuring and abstracting what is known, but also on representing uncertainty and knowledge gaps explicitly.

4.1 Spatial Knowledge Representation for Mobile Cognitive Systems

Before designing a representation of spatial knowledge, it is important to review the aspects a representation should focus on. In this work, we focus on mobile cognitive systems. Based on the analysis of existing approaches [10, 9, 29] as well as the ongoing research in CogX on artificial cognitive systems, we have identified several areas of functionality, usually realized through separate subsystems, that must be supported by the representation. These include localization, navigation, and autonomous exploration, but also understanding and exploiting semantics associated with space, human-like conceptualization and categorization of space, reasoning about spatial units and their relations, human-robot communication, action planning, object finding and visual servoing, and finally recording and recalling episodic memories.

Having in mind the aforementioned functionalities, aspects covered by a representation of spatial knowledge as well as limitations resulting from practical implementations, we have designed a representation having the following properties.

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 [11]. 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, re-

dundancy is avoided and whenever possible and affordable, new knowledge should should be inferred from the existing information. It is important to note that uncertainties associated with represented symbols should are explicitly modeled.

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 [31]). 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 to make planning and higher level conceptualization tractable.

A representation should allow not only for representing instantiations of spatial segments visited by the robot. It is equally important to provide means for representing unexplored space. In our representation, unexplored space (which is a gap in spatial knowledge) is explicitely modeled. Furthermore, categorical knowledge is represented that is not specific to any particular location and instead corresponds to general knowledge about the world. Typical examples are categorical models of appearance of places [25], objects [23] or properties of space recognized by humans (e.g. shape, size etc.).

Finally, we focus on the fundamental role of the representation in human-robot interaction. The representation models correspondence between the represented symbols and human concepts of space. This correspondence can be used to generat and resolve spatial referring expressions [30] as well as path descriptions.

4.2 Structure of the Representation

Figure 6 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. The properties characterizing each layer is summarized in Table 1. 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-

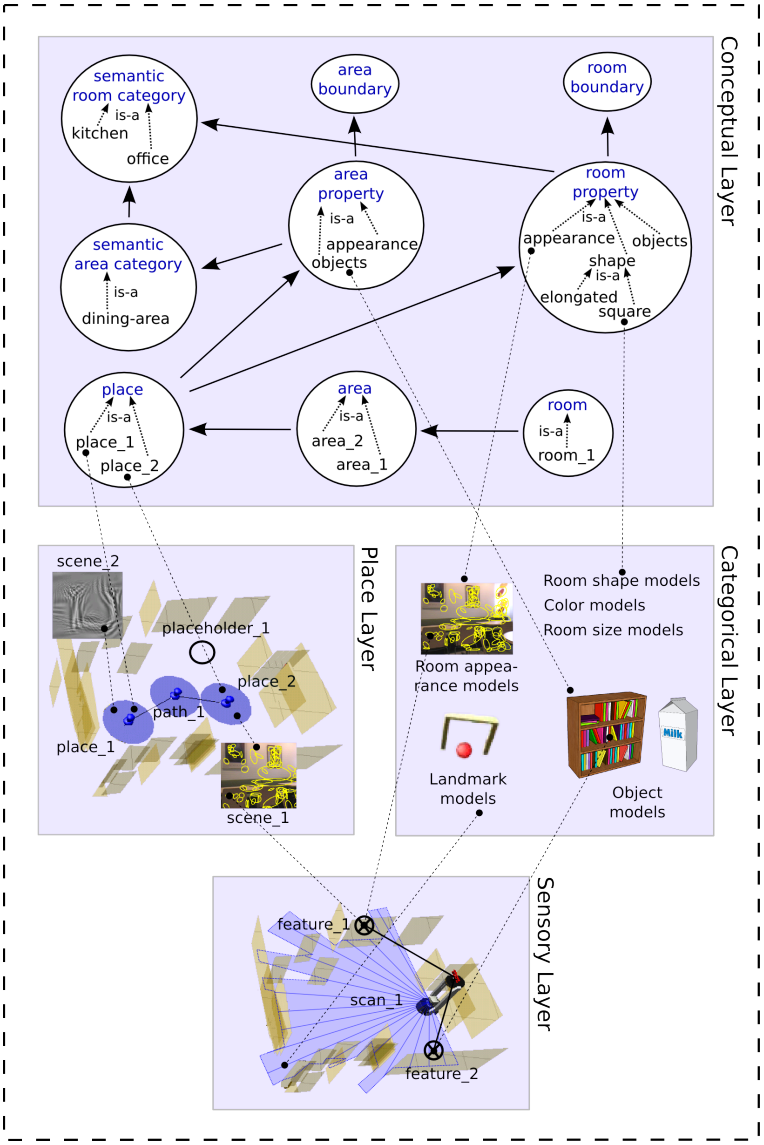


Figure 6: The layered structure of the spatial representation. The position of each layer within the representation corresponds to the level of abstraction of the spatial knowledge.

Property	Sensory Layer	Place Layer	Categorical Layer	Conceptual Layer
Aspects represented	Accurate geometry and appearance	Local spatial relations, coarse appearance, geometry	Perceptual categorical knowledge	High-level spatial concepts / Links concepts $\leftrightarrow$ entities
Agent's position	Pose within the local map	Place ID	Relationship to the categorical models	Expressed in terms of high level spatial concepts
Spatial scope	Small-scale, local	Large-scale	Global	Global
Knowledge persistence	Short-term	Long-term	Very long-term	Life-long / Very long-term
Knowledge decay	Replacement	Generalization, forgetting	Generalization	None / Forgetting
Information flow	Bottom-up	Primarily bottom-up	Primarily bottom-up	Top-down and bottom-up

Table 1: Comparison of properties of the four layers of the spatial representation.

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 [27].

4.2.1 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 [28, 12]. 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 guiding the attentional process, may be imposed from upper layers. It can contain data of both a 2D and 3D nature.

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. This can be useful for providing "virtual sensing" such as 360° laser scans based on short-term temporal sensory integration as well as generation of features based on spatial constellations of landmarks located outside the field of view of the sensor. Additionally, it could be used for temporal filtering of sensory input or providing robustness to occlusions. 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 addition to the landmark based representation, *local gridmaps* are maintained, centered on each Place. These are metrical representations of explored and unexplored space, covering a single Place and its immediate surroundings. Figure 7 on page 23 shows two examples of local grid maps of adjacent Places. The white area is "known empty" space: open areas swept out by the robot's laser scanner from within the Place. The black regions represent obstacles, and gray denotes unexplored space, which constitutes a knowledge gap. The local grid maps are used to generate placeholders, which allow for exploratory behavior (see below).

The visual search routine maintains hypotheses about existence of objects of specific categories at specific locations using a probabilistic grid representation [3]. The probabilistic grid representation is shaped based on multiple cues providing evidence about the existance of the particular object class. Examples of such evidence is the presence of occupied space which may either be an observation of the object itself or a supporting structure for the object. Another example is 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 AVS. The first one is unexplored space, described above, representing a gap in that the structural information of space 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. This is represented explictly in a grid with the same dimensions as the probabilistic grid holding the object location probability.

4.2.2 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.

Each place is defined in terms of features that are represented in the sensory layer, but also spatial relations to other places. 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. Similarly, 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.

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 **??** on page ??. 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, described later in Section **??** on page ??. The measures used are the *coverage estimate* (CE) and the *frontier length estimate* (FLE), cf. Figure 8 on the following 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.

The place layer operates on distinct places as well as their connectivity and spatial relations to neighboring places. No global representation of the whole environment is maintained. Still, since the local connectivity is available, global representation (e.g. a global metric map) can be derived

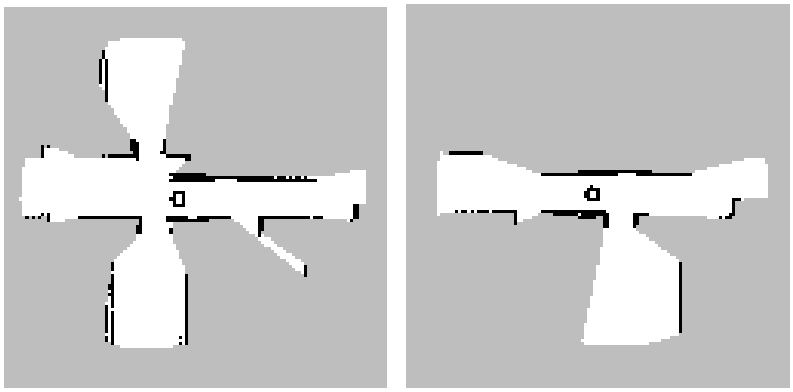


Figure 7: Example of exploration grid maps

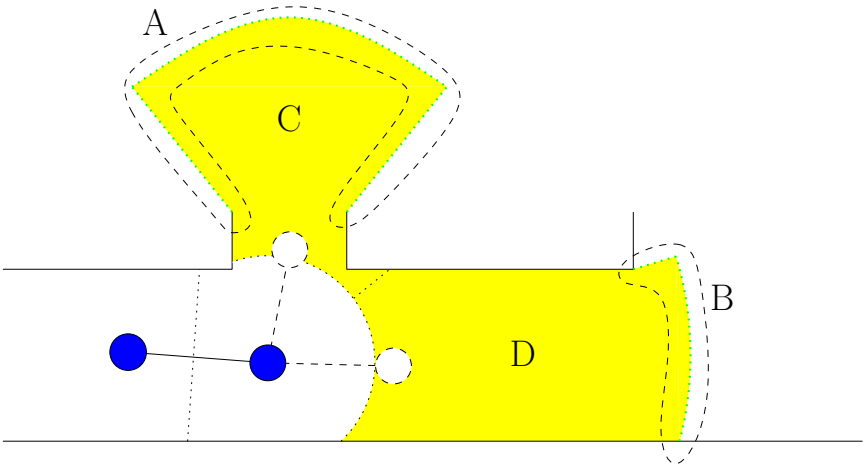


Figure 8: Placeholder creation. Dashed circles are placeholders, each representing one placeholder. A and B are frontier length estimates, C and D are coverage estimates for the respective placeholders.

when needed. This representation will not be precise, but will preserve the connectivity and relaxed spatial relations between all the places.

4.2.3 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

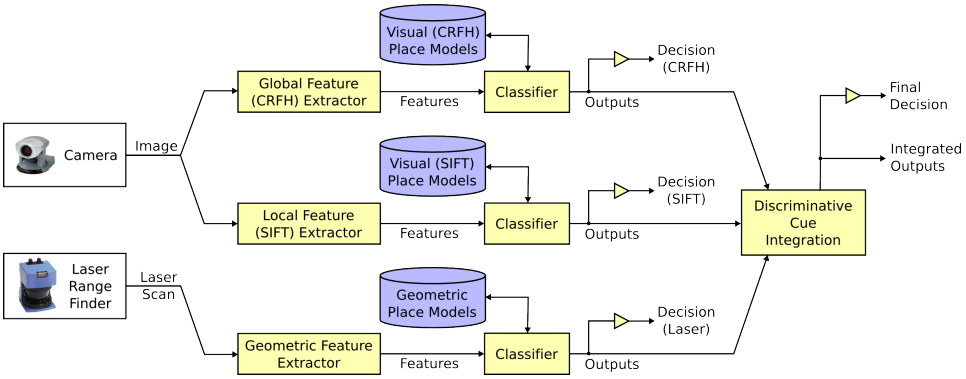


Figure 9: Architecture of the multi-modal place classification system.

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. Due to the high complexity of the models, unused knowledge might be compressed and generalized.

The architecture of the place classification system [26] is illustrated in Figure 9. The system relies on multiple visual cues corresponding to different types of image features as well as simple geometrical cues extracted from laser range scans [20]. In particular, we employed local features based on the SIFT descriptor [19] and global descriptor based on the Composed Receptive Field Histograms [16]. The cues are processed independently. For each cue, there is a separate path in the system which consists of two main building blocks: a feature extractor and the SVM classifier [**?**]. Each classifier relies on an independent model trained on a single cue and produces a set of outputs indicating its decision and the uncertainty associated with that decision [24]. These outputs can be used directly to obtain the final decision separately for each cue. In cases when several cues are available, the single-cue outputs are combined using a high-level discriminative accumulation scheme producing integrated outputs from which the final decision is derived. Since each of the cues is treated independently, the system can decide to acquire and process additional information only when necessary e.g. only in difficult cases. This scheme is referred to as Confidence-based Cue Integration [24].

The visual search routine uses the object recognition method proposed in [22] 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 [3].

4.2.4 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 6 on page 19, cf. [31]. 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

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

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 6 on page 19) 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.

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

5 Beliefs about situated dialogue

A gap in dialogue typically is seen as a lack of understanding, leading potentially to a breakdown in communication. Clarification mechanisms can help to resolve this. Several discourse theories identify levels at which such gaps can arise. This can go all the way down to problems in situationally grounding dialogue. In this paper, we follow out the viewpoint that if we want to investigate self-introspection and self-extension in situated dialogue processing, we need to look both at representation and processing. We describe how various levels of representing meaning can make gaps in their interpretation explicit, and available for self-introspection. Given a gap, we show how a robot can act upon it, driving behavior to extend its knowledge, ultimately to adapt its processing models so as avoid making the same gap twice.

5.1 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 [7]. This can go all the way down to problems in situationally grounding dialogue. We explored the latter problem in more detail in [15]. 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 §5.2. 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 §5.3 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.

5.2 Logic forms for representing meaning

We represent linguistic meaning as an ontologically richly sorted, relational structure. This structure is a *logical form* [14, 4] in a decidable fragment of modal logic [5, 2]. The modal-logical aspect of the logic makes it possible to build up structures using named relations. A novel construct, called a "nominal" [5] 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.

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

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 [21, 6]. Figure 10 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 [17, 18].

5.3 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,

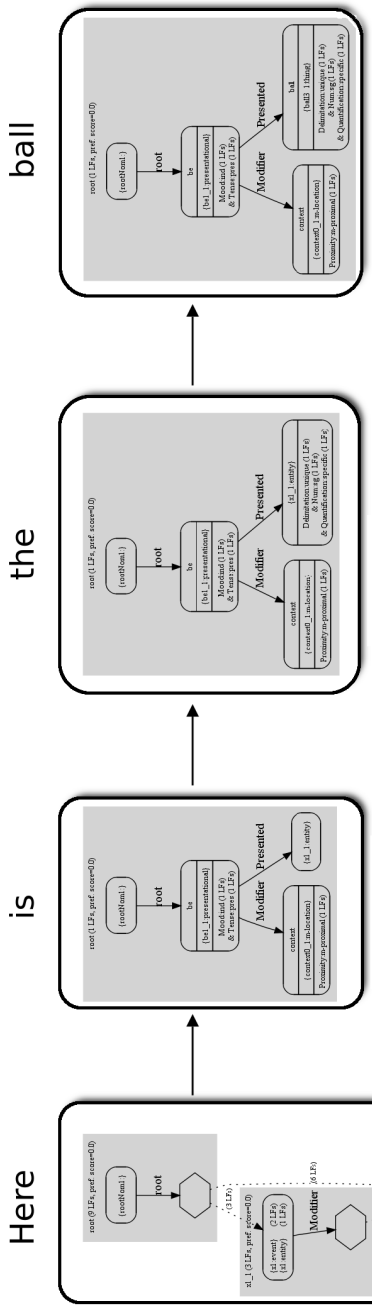


Figure 10: Example of incremental parsing and packing of logical forms, for the utterance "Here is the ball". Four steps are shown.

we present an approach to how situated beliefs are formed. We exploit the 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 [**?**]) 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.

5.3.1 Attaining and maintaining common ground

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 [**?**], i.e. that they are truthful and only say what they believe to be true and for what they have evidence.

5.4 Conclusions

Looking back at the discussion in this paper, 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 showed here the representational side of self-understanding and self-extension, with regard to situated dialogue processing. In WP6, we look at how we can take the self-understanding aspect in representation then to the self-extending aspect in processing, to yield adaptive dialogue processing.

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 happen to be the key ingredients.

Information obtainable by visual methods will often be incomplete (occlusions, backsides of objects, objects missed alltogether) or inaccurate (uncertainties with regard to object identity or pose).

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 types of knowledge gaps regarding scene interpretation:

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

In the current architecture the former is addressed by a 3D attentional mechanism (plane pop-out) which produces 3D spaces of interest (SOIs) that serve as object candidates. The latter is done by associating properties of SOIs (colour, shape) with learned labels or recognising learned object instances. Finally, once we have a hypothesis about location and identity of an object we can track it over time.

Observation Functions

Each detector (planes, SOIs, objects) has some associated confidence c. Using labeled training data we learn observation functions for each detector: $P(c \,|\, object = true)$. This will produce a monomodal distribution (if not the confidence measure is chosen badly) to be expressed e.g. as a Gaussian. This allows using disparate confidence values which are not themselves probabilities within a probabilistic framework.

6.1 Where?

The first type of gap "Where are objects?" or to be more precise "Where is a high likelihood of objects?" is addressed by bottom-up 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. SOIs extracted from 3D stereo data are further refined by back-projection onto the 2D image and colour-based segmentation.

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

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

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 as a next step we will learn observation functions using labeled training data

$$P(\rho \mid plane = true)$$

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 centers 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 labeled training data we can then learn an observation function

$$P(c \mid soi = true)$$

6.1.3 Proto Objects

SOIs are object candiates which are segmented from their supporting planes. The quality of the segmentation depends on texture available for stereo reconstruction and often this segmenation 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 backproject 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 backprojected SOI and the segmentation mask the corresponding refined foreground/background labelling. Foreground points is the refined list of 3D points corresponding to the precise segmentation mask. Furthermore the proto object contains a reference to its soure 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.

6.2 What?

The second type of gap "What are the properties of objects?" is addressed by (1) incremental learning and recognition of visual properties as well as (2) learning and recognising object shape and identities. The former is addressed in Section 11about cross-modal beliefs.

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
  pose
  // identity
  identityLabels
  identityDistribution
  // 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 continous probability density distribution.

6.3 Recognition

Object instance recognition is based on SIFT features, which are mapped onto the surface of an object during the learning phase. SIFT descriptors are then used to build a codebook, where SIFT 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.

To distinguish between hallucinating false detections and correct object locations we define the object confidence

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

of detection 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 proper probability however so we will again learn observation functions

$$P(c \mid object = true)$$

6.3.1 Detection

6.4 Tracking

7 Planning: how beliefs change under action

8 Representations in Planning

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 STRIPS planning [], that model 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) *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 for gathering perceptual evidence. Concretely, thus far 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 Markov Decision Process[5] for

[4] Planning Domain Definition Language.
[5] Markov Decision Process.

modelling uncertainty in actions and state, and the POMDP[6] for modelling uncertainty in perceptions/observations. In the sequel we review those in turn, and then motivate a more expressive variant of PDDL, called DT-PDDL,[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. Making all the above ideas concrete, we describe a simple version of the CogX *dora* scenario using DTPDDL.

8.1 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 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

[6] Partially Observable Markov Decision Process.

[7] Decision-Theoretic PDDL.

is PSPACE-complete in general, and NP-Complete under fairly reasonable restrictions to plan length.

8.2 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 [**?**]. The 3rd Planning Domain Definition Language (PDDL3.0 – pronounced "pea-diddle") is the language of choice of the planning community [**?**, **?**, **?**]. 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 make the ideas of the previous section and Appendix A by given an example from a familiar CogX scenario.

8.2.1 Example: Classical 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 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 effect 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 the equality – i.e., "(= ...)" predicate. 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 `widgets`, `features`, and `places` and their associated `labels` are all objects. We also introduce a `model` type, instances of which are either `widgets` or `features`.

Following a description of the model types, we can describe the relations and concepts that further exists withing the world. In particular, we give the *:predicates* and state functions *:s-functions*. All of these we 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 are a variable name. Those are included in the predicate specification here for pedagogical reasons. 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:

`move-to-connected-place`(`Hall`, `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 that 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`.

9 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. Here there is quantified uncertainty about the state of the environment, and also the robot can only gather evidence, via perception, about the **true** environment.

9.1 Markov Decision Processes (MDPs)

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

and [**?**].

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:

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

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 $\mathsf{R} : \mathcal{S} \times \mathcal{A} \to \Re$. 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 $\pi : \mathcal{S} \to \mathcal{A}$ 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 $\pi : \mathcal{S}^* \to \mathcal{A}$.[8]

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.

[8] 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.

$$V_\pi(s) = \lim_{n\to\infty} \mathsf{E}\left[\sum_{i=0}^{n} \beta^i \mathsf{R}^\pi_{\mathsf{seq}}(\Gamma_i) \mid \pi, \Gamma_0 = s\right] \tag{9}$$

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:

$$V_\pi(s) = R(s, \pi(s)) + \beta \sum_{s' \in \mathcal{S}} \mathsf{Pr}(s, \pi(s), s') V_\pi(s') \tag{10}$$

In (Eqns 9 and 10) $0 < \beta < 1$ 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.,

$$V_{\pi^*} = T(V_{\pi^*}) \tag{11}$$

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

$$\|T(V_1) - T(V_2)\|_\infty \leq \beta \|V_1 - V_2\|_\infty \tag{12}$$

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

$$\lim_{n\to\infty} T^n(V) = V_{\pi^*} \tag{13}$$

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

9.2 Partially Observable Markov Decision Processes (POMDPs)

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 $\mu : \mathbb{O}^* \rightarrow P_{\mathcal{A}}$ where $\mu_a(o_0, .., o_t)$ is the probability that we execute action a given observation history $o_0, .., o_t$.[9] 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:

$$V_n(s) = \sum_{a \in \mathcal{A}} \psi_n(a) \mathsf{R}(s, a) + \beta \sum_{a,o,s',n'} \eta_n(a, o, n') \mathsf{Pr}(s, a, s') v_o(s', a) V_{n'}(s') \tag{14}$$

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:

$$V_{\mathtt{FSC}}(b) = \max_{n \in \mathcal{N}} \sum_{s \in \mathbb{S}} b(s) V_n(s) \tag{15}$$

9.3 DTPDDL

DTPDDL extends PPDDL in four key respects. First, the effects of actions can be stochastic, and therefore the language facilitates the specification of actions 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 PPDDL for problem definitions did not support

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

specification of a problem start state distribution – i.e., full observability was assumed, and hence the starting state was fully observable. For DT-PDDL we have introduced syntax for compactly specifying the starting state distribution.

9.4 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* must commit to the *true* location of the `cornflakes`. If she makes the correct commitment on that point, then she is rewarded strongly, and otherwise she is punished terribly. Thus, she has an interest in disambiguating through 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 model 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 `place`s have particular `label`s, and that a `place` features a `model` or not. 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 rather 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 given 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 `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 `type`s 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 so 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:[10]

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

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* can not observe elements in "*s-functions*" and "*:predicates*" description elements unless they are known with certainty according to her belief-state distribution. However, she can

[10]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.

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:

$$(\textit{observed\_model\_at\_place}\ \texttt{Kitchen}\ \texttt{Cornflakes}) \tag{16}$$

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`, then with probability:

`probability__observe_feature_at_place_with_label__if_true` ?`location` ?`model` ?l)

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

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)
              )
             )
           )
          )
```

10 Representations for Continual Planning

The representation used by the continual planning system is based on the SAS$^+$ formalism [**?**]. 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 [**?**]. 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 $\mathcal{E}$.[1]

1) This describes a "grounded" domain. Maybe we should be even more general. Cf. Notes section.

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:

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

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 next 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 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

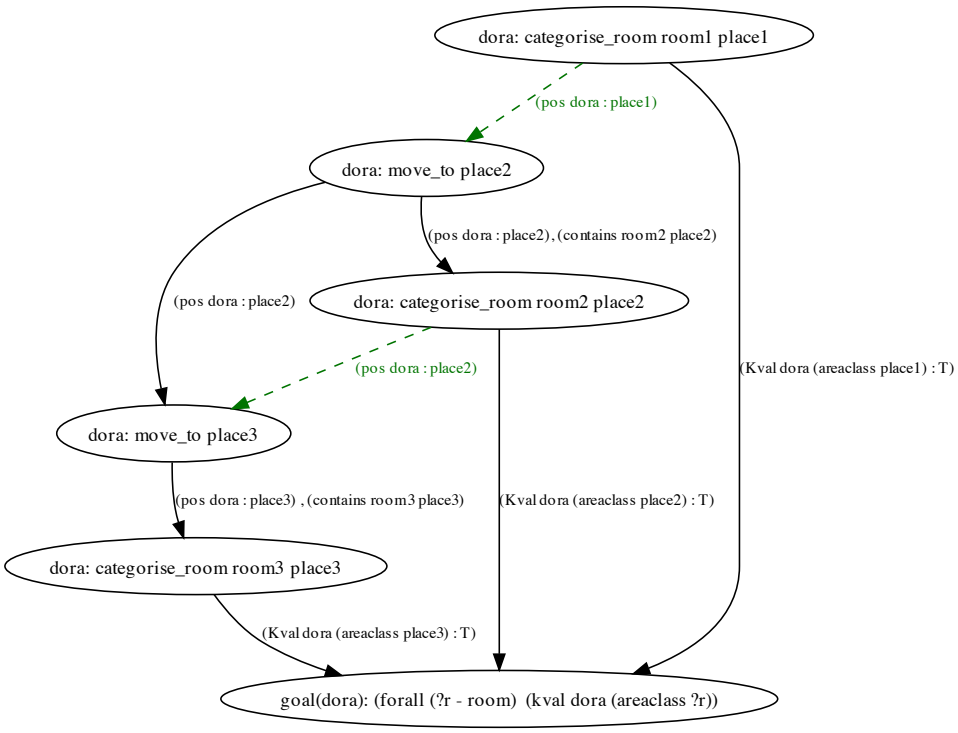


Figure 11: A plan using sensory actions to satisfy epistemic goals

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" ($Kval^{R,H}_{pos(cornflakes)} = \perp^H$) and "R does not know if H knows the location of the cornflakes" ($(Kval^{R,H}_{pos(cornflakes)} = \perp^H$).

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).

11 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.

11.1 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) [**?**] 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 [**?**]. Specifically, the oKDE is defined by the so-called model of the observed samples $\mathbf{S}_{\text{model}}$,

$$\mathbf{S}_{\text{model}} = \{p_s(\mathbf{x}), \{q_i(\mathbf{x})\}_{i=1:N}\}, \tag{17}$$

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[11]. The *sample distribution* is modelled by a mixture of Gaussians

$$p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x} - \mathbf{x}_i), \tag{18}$$

where

$$\phi_{\mathbf{\Sigma}}(\mathbf{x} - \mu) = (2\pi)^{-\frac{d}{2}} |\mathbf{\Sigma}|^{-\frac{1}{2}} e^{(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\mu))} \tag{19}$$

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):

$$p_{\text{KDE}}(\mathbf{x}) = \phi_{\mathbf{H}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}+\mathbf{\Sigma}_{si}}(\mathbf{x} - \mathbf{x}_i). \tag{20}$$

[11] we will not go into details about the detailed model here, since it exceeds the scope of this document and we refer to [**?**] 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.

Note that the online KDE from (20) 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.

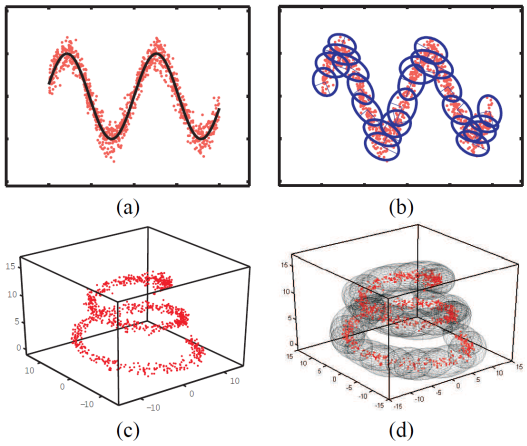
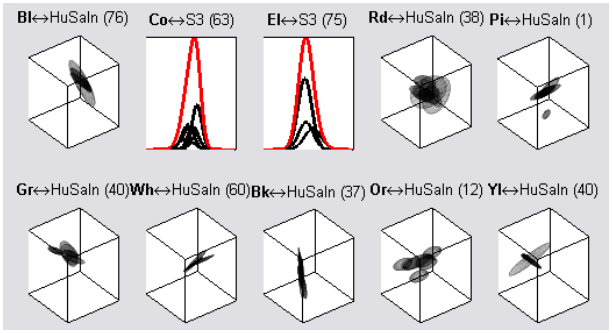


Figure 13: First row shows the sinusoidal distribution and the second row shows the spiral distribution. Left column shows the reference distributions and the right column shows the estimated distribution using oKDE after observing a 1000 samples.

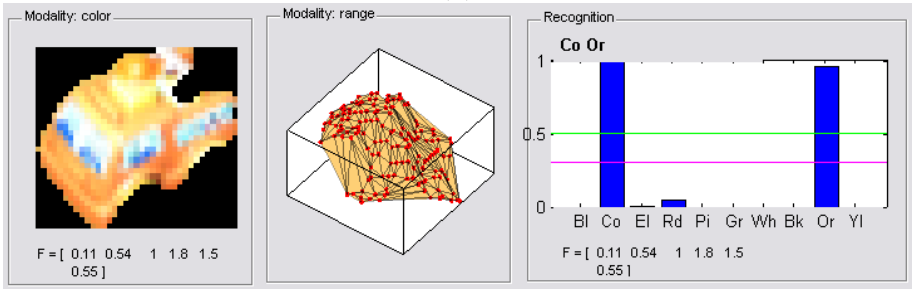
The continuous learning proceeds by extracting the visual data in a form of a highdimensional features (e.g., multiple 1D features relating to shape, texture, color and intensity of the observed object) and oKDE is used to estimate the pdf in this high-dimensional feature space. In this respect, the 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 [**?**]. For the exact and detailed description of the algorithm for feature selection algorithm, see [**?**], Algorithm 1.

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.

11.1.1 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:

$$p(z|s) = p(z|m_k, s)p(m_k|s) + p(z|m_u, s)p(m_u|s). \tag{21}$$

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,

$$p(z|m_k, s) = \sum_{i=1}^{K} p(z|M_i, m_k, s)p(M_i|m_k, s). \tag{22}$$

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 (21) becomes

$$\begin{aligned} p(z|s) = p(m_k|s) \sum_{i=1}^{K} p(z|M_i, m_k, s)p(M_i|m_k, s) \\ + p(m_u|s)p(z|M_0, m_u, s)p(M_0|m_u, s). \end{aligned} \tag{23}$$

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

$$p(M_i|z, s) \propto p(z|M_i, m, s)p(M_i|m, s)p(m|s), \tag{24}$$

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 [**?**], 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[12]:

$$p(m_\mathrm{u}|s) \quad = \quad e^{-0.5(\frac{N}{\sigma_N})^2}. \tag{25}$$

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

11.1.2 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

[12]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.

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 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.

12 Conclusion

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.

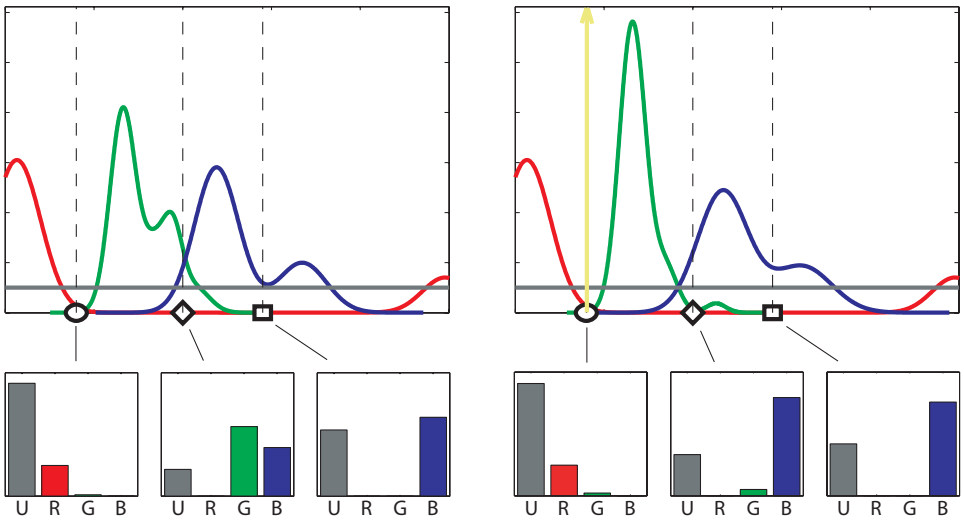


Figure 15: Example of detecting the knowledge gaps and updating the 1D KDE representations. Top row: probability distributions for three colours (red, green, blue lines) and unknown model (gray line) in 1D feature space. Bottom row: a posteriori probabilities for the unknown model (U) and three colours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

13 Annexes

13.1 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.

13.2 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.

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.

2.3 Example of Markov Logic Network

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.

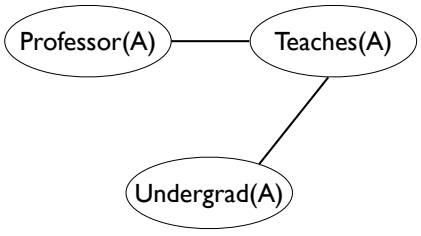


Fig. 1. Example of ground Markov Network $M_{L,C}$ given the Markov logic network $L = \langle(\texttt{Professor(x)} \rightarrow \texttt{Teaches(x)}, w_1), (\texttt{Undergrad(x)} \rightarrow \neg\texttt{Teaches(x)}, w_2)\rangle$ and the constants $C = \{A\}$. An edge between two nodes signifies that 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.

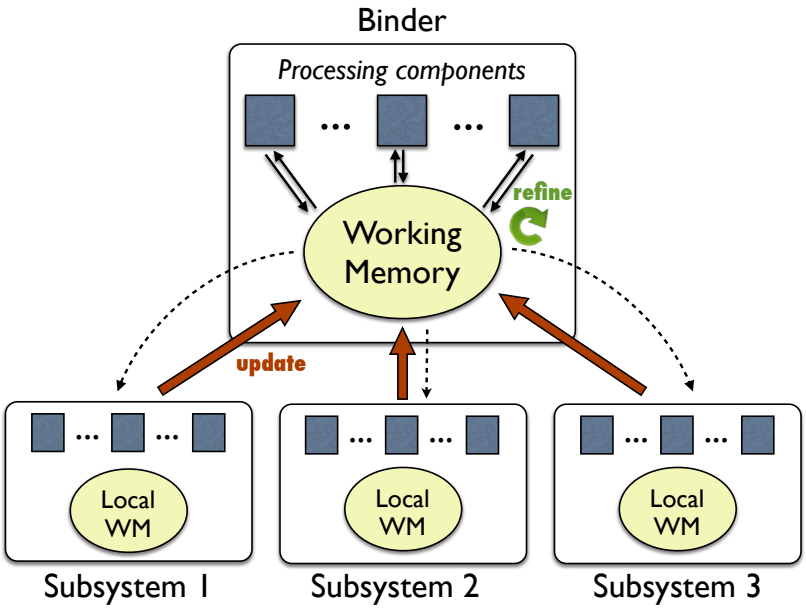


Fig. 2. Schema of the cognitive architecture in relation with the binder

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.

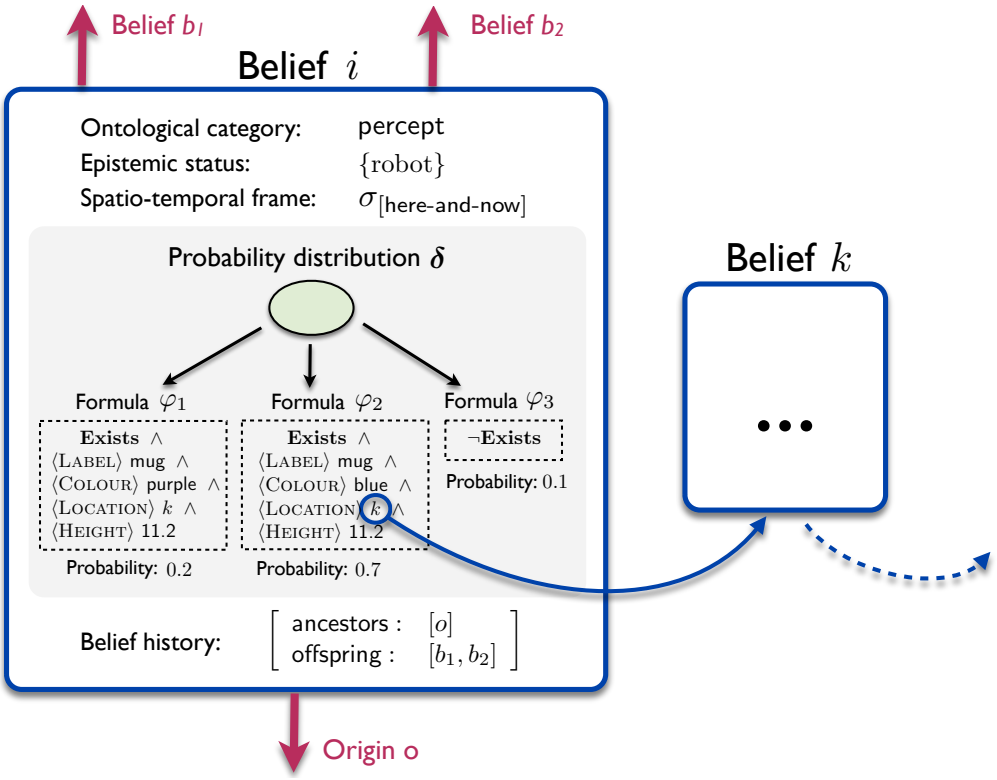


Fig. 4. Schematic view of a belief representation.

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

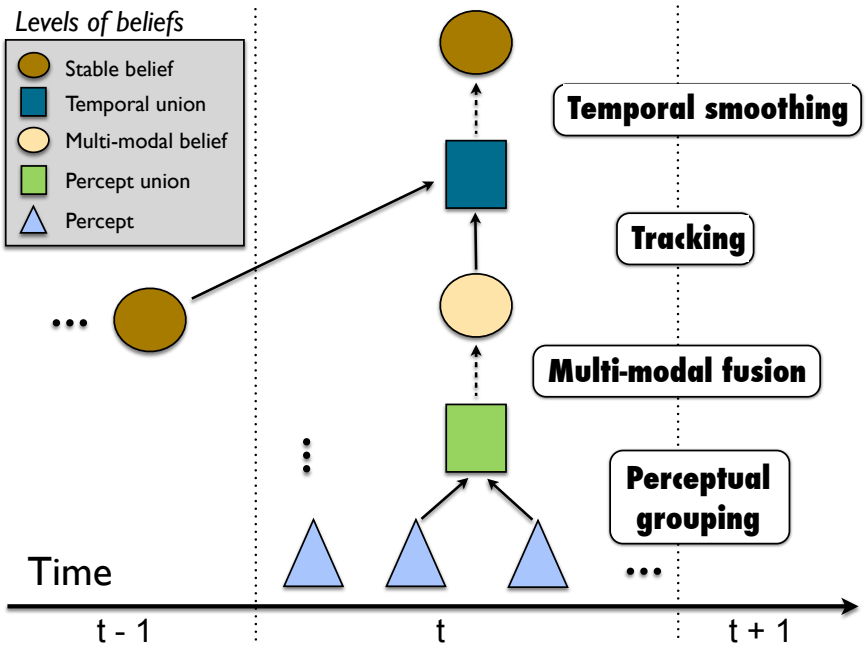


Fig. 5. Bottom-up belief model formation.

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.

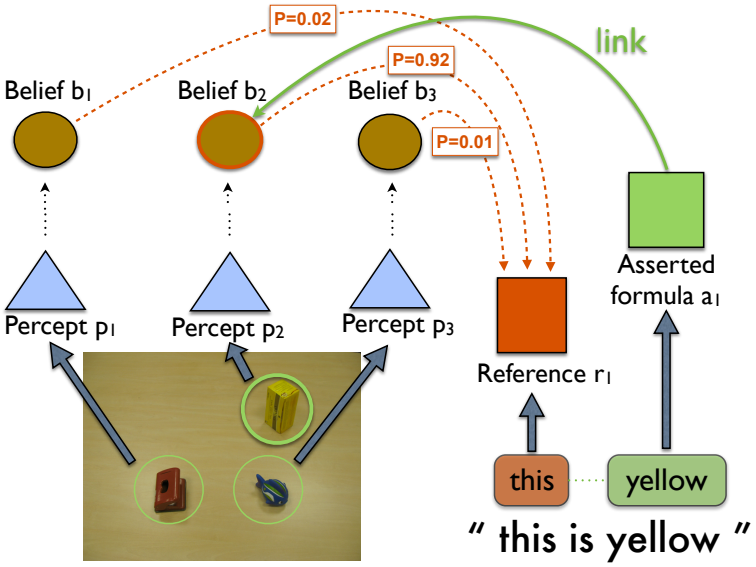


Fig. 6. An utterance such as "this is yellow" illustrates the two mechanisms of referencing and belief extension. First, the expression "this" is resolved to a particular entity. Since "this" is a deictic, the resolution is performed on basis of saliency measures. The belief $\mathtt{B}_2$ is selected 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 14th 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.

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 [**?**] – to contain syntactic elements for describing domains and corresponding problems that feature partial observability. Our work draws heavily on the work of [**?**] 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))
)
```

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] 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.

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

[6] 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.

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

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

[9] COGNIRON: The cognitive robot companion. Website: `http://www.cogniron.org`.

[10] CoSy: Cognitive Systems for Cognitive Assistants. Website: `http://www.cognitivesystems.org/`.

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

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

[13] 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.

[14] 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.

[15] 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.

[16] O. Linde and T. Lindeberg. Object recognition using composed receptive field histograms of higher dimensionality. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, Cambridge, UK, 2004.

[17] 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.

[18] 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.

[19] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[20] Oscar Martínez Mozos, Rudolph Triebel, Patric Jensfelt, Axel Rottmann, and Wolfram Burgard. Supervised semantic labeling of places using information extracted from laser and vision sensor data. *Robotics and Autonomous Systems Journal*, 55(5):391–402, May 2007.

[21] 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.

[22] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, accepted, 2009.

[23] A. Pinz. Object categorization. *Foundations and Trends in Computer Graphics and Vision*, 1(4):255–353, 2006.

[24] 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.

[25] 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.

[26] Andrzej Pronobis, Oscar M. Mozos, Barbara Caputo, and Patric Jensfelt. Multi-modal semantic place classification. *The International Journal of Robotics Research (IJRR)*, 29(2-3):298–320, February 2010.

[27] 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.

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

[29] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R Murphy, editors, *AI-based Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, 1998.

[30] Hendrik Zender, Geert-Jan M. Kruijff, and Ivana 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.

[31] 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.