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

DR 1.5: Unifying representations of gaps in knowledge

Jeremy L. Wyatt, Nick Hawes, Danijel Skočaj, Alen Vrečko, Moritz Goebelbecker, Thomas Keller

⟨`cogx@cs.bham.ac.uk`⟩

Due date of deliverable:	May 30 2012
Actual submission date:	May 30 2012
Lead partner:	BHAM
Revision:	final
Dissemination level:	PU

CogX is concerned with establishing a theoretical framework within which to pose different problems related to self-understanding and self-extension. In this paper we describe the final state of our approach to representing and filling gaps and uncertainties in the robot's knowledge state. Rather than provide an exhaustive survey of the ways we choose to represent gaps in specific modalities (as already performed in DR.1.2 in year 2), we instead show for a small number of different types of gap how they are represented and filled. The key contribution is the difference in how task-driven gap filling and curiosity-driven gap filling occurs. We show how these two types can be unified through a hierarchy of drives in our motivation system, and how we detect, select and fill gaps in each case.

Executive Summary

This report presents the work during the CogX project on representations of gaps and uncertainty. In particular it describes our systems theory of how such gaps and uncertainties should be dealt with. Earlier versions of this theory have been presented, e.g. in [6]. In this paper we extend that theory to describe more fully how an agent can reason about gaps, and how to fill them. This required us to solve several open problems in planning and knowledge representation. The report is split into parts focused on *curiosity driven knowledge gathering* and on *task driven knowledge gathering*. During the former the robot must compare quite different learning goals and select one. This requires that the robot is able to perform introspection on its own representations, and we show how we do this in the George system. Our other assumption here is that the comparison of different possible learning goals should be done rapidly via a motivation system because of the intractability of planning the effects of all possible learning activities. The motivation system makes the problem tractable by heuristically comparing goals and choosing a small subset to achieve, rather than reasoning about plans to achieve many of those goals. Regarding task driven knowledge gathering our idea is that by enabling a planner to reason about (limited) open worlds the robot is able to hypothesise the existence of unsensed (but previously experienced) entities that are useful or necessary to achieving its task. In addition it is able to explain planning failures (which form one of the kinds of surprise we outlined at the beginning of the project) in the same way. In this paper we describe all our of task driven work via a case study from the Dora system, and the curiosity driven work via a case study from the George system. Finally task driven and curiosity driven knowledge gathering can both be handled by our motivation system, giving a simple way to choose between them. In DR.1.4 we will describe a more sophisticated way to handle the trade-off between curiosity driven and task driven activity. The task addressed in DR.1.5 is Task 1.1, of which we said

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

Role of gaps in CogX

Representations of gaps and uncertainty in knowledge have been central to CogX. The challenge was how to deal with these gaps in a general way useful for planning and goal management. The approach in CogX is representationally driven. In this report we outline how the representations of gaps we developed, and how the planning and motivation frameworks that we developed are unified into an overall approach.

Contribution to the CogX scenarios and prototypes

In this report we have described our overall approach to representing and filling different types of knowledge gaps. The relation to the CogX scenarios is clear. We have used the George system as a way of exploring methods for curiosity driven gap filling, and Dora as a platform for task driven gap filling. Thus the methods described in this report relate very clearly to one or other system.

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.

This deliverable also contributes to the following project objectives:

O1 A unified framework for representing beliefs about representations of action effects, observation models, incomplete information and categorical knowledge.

O3 Representations of how actions will alter the belief state of the cognitive system, and those of other agents, as represented in the first two objectives, i.e. models of the effects of action on beliefs about space, categorical knowledge, action effects, dialogue moves etc.

The main objective for us by this stage was to create unified representations of gaps. In fact, we achieved this rather early in the project (see DR.1.2) and now we have tried to show how the reasoning process about how the gaps are filled is linked to these representations of gaps. Thus we have looked for a systems level theory of how gaps are represented, reasoned about and filled that covers both curiosity driven and task driven activity. Parts of this theory are architectural and parts are based on ideas about how planning can be performed in open worlds that include beliefs in their state description. This theory is therefore an important part of our overall approach to self-understanding and self-extension.

1.2 Actual work performed

The report describes work on the George and Dora systems performed between years 2 and 4 of the project. It specifically covers the use of the motivation and planning systems to represent and choose gaps to fill. We have posed this problem separately for curiosity driven and task-driven gaps, but here we also describe how a motive management system can be simply used to allow both to be handled.

2 Introduction

How should an agent reason about things it might learn about, or information it might discover during performance of a task? This is a hard problem, but one that is necessary to solve in order to build systems that are robust to novelty, uncertainty and change. In this paper we outline the approach taken in two robot systems to two different versions of the problem. We refer to these two versions of our overall problem as *curiosity driven* and *task driven* knowledge gathering respectively. In curiosity driven knowledge gathering the robot must identify possible learning goals, choose one, and then reason about how to achieve that learning goal. In task driven knowledge gathering the robot must represent things it must know in order to be able to perform its task, and if it doesn't already know them, it must reason about how they can be acquired. We show how each problem can be posed, and how these two problems can in principle be handled by a single system. There are several ideas that underpin our theory, and these are outlined now. It is important to note that here we are not recapitulating in this summary all of the details of the theory, but giving a qualitative overview of the main ideas.

Gaps and uncertainties in knowledge. A robot that is to extend its knowledge will benefit from representing what it knows and doesn't know. Specifically in our case that means representing items that are known to be in the world, but also items that are not known to exist. These unknown items could include concrete entities such as objects, places or rooms. But they can also include abstract entities such as qualities, including colours, shapes and appearances of objects, places or rooms. It may be necessary for the robot to express the fact that it doesn't know at the moment what type of place somewhere is, or what a particular colour might look like. We refer to these unknowns as *uncertainties* and *gaps*. Uncertainties refer to closed worlds, i.e. where there are a fixed and known number of classes, but where there is ambiguity about which class or category an entity belongs to. This includes situations such as when the colour of an object is known to be from one of several defined classes, but it is not known which. A knowledge gap refers loosely to any situation which is not easily captured as a closed world. For example a gap exists when an entity is not from a known set of categories, such as when an object is of a novel visual category. Second it can be when an instance of a known category is not known to exist in this particular world, e.g. if a type of room is not in our map. Gaps require the ability to reason about open worlds. These notions of gaps and uncertainties are blurred: there is a boundary when there is some likelihood the entity is from a known class, and some that it is from a novel class.

Open worlds and epistemic action effects. In our approach the robot plans its learning and information gathering actions. This requires that it is able to reason about the effects of such actions on the knowledge gaps and uncertainties it has. This in turn requires that we are able to solve several problems in planning. First the robot must be able to reason about entities it hasn't seen or facts it doesn't know as when operating under incomplete knowledge it is often necessary to generate and use hypotheses about what might be the case in order to be able to formulate a plan. This could include the creation of symbols denoting hypothesised objects, rooms, relations or properties. To do this our robot will require an ontology that includes possibilities. We broadly use approaches based either on probabilities or modal logics. Second the robot must be able to represent, at plan time, the effects of the actions in its plan on gaps its knowledge, i.e. the planner's action models must represent the epistemic effects of actions. Where the number of specific knowledge outcomes is very large, or where new information might be acquired (such as learning the name of a new colour) this may be best achieved by representing the type of knowledge accquired, rather than the specific knowledge outcomes themselves. We have previously created an extension to PDDL that allows this, and in particular allows us to represent the likelihoods of different epistemic effects of actions.

Introspection and gap detection In our framework it is necessary that for each type of knowledge there is a way to detect a gap where one is not obvious. This requires that the robot has the ability to introspect on its representations to spot gaps. The algorithms for introspection depend on the underlying representations. This requires that modality and representation specific procedures are often employed for gap detection, but there are some general principles according to which these should work. It is useful to be able to produce a measure of how well observations are explained by each possible class of object or room, so that a judgement may be made as to whether the observation is caused by some previously unexperienced class of object, place or event. Alternatively the same kind of measure can be used to quantify the degree of uncertainty about which of a known set of class values (e.g. known colours) causes the observed data. Finally introspection and abduction are useful to generate hypotheses to explain otherwise unexplained data, or to better explain it. We will handle this last type of introspection in a planning framework as described next.

Surprises, failures and background knowledge. When a plan is created and then executed it may be the case that the plan does not execute in the way expected. In such situations either positive or negative surprises occur. A positive surprise is when an event or entity has additional features that were unpredicted, e.g. when an item is found in a previously unexpected place. Such a positive surprise provides a learning opportunity in that possible explanations at different levels of generality may be applied, and these hypotheses must be generated and tested. A negative surprise is when an entity fails to appear as expected, e.g. the item looked for is not present in the place expected, or the colour of an item is different from that expected. This second case can lead to the need for unlearning of previously learned knowledge. In a task driven case it can also lead to an execution failure, and if no alternative plan is to be found then this constitutes a planning failure. We handle such a failure by a planning process in which additional knowledge (which we call background knowledge) is brought to bear, and where the agent again has the ability to plan in a limited open world, i.e. to use this additional background knowledge to create hypothesised additional objects and properties to allow a new plan to be formulated. Postulating these additional entities is one way of explaining the surprise, and the plan tests these explanations while achieving the original task. It can sensibly be asked why we should not use the additional knowledge in the initial planning process: for then there could be no surprises. The answer is simply that to achieve scalability to a very large background knowledge base – as will be needed in more general purpose robots – it is simply not possible to plan efficiently with all possible knowledge. Planning will need to use only a small subset of foregrounded knowledge to achieve any particular plan. A difficult open question is how a robot can efficiently decide what that foregrounded knowledge should be.

Motivation and efficiency. If a learning robot can generate many different learning goals that it could pursue that it is not feasible to plan for all of them. In addition, in a real-time system such as a robot operating in an environment new learning opportunities will arise as existing ones are being pursued. In addition human generated tasks must be dealt with as they arise. We have developed a theory where learning opportunities are raised as motives (similar to desires), and are prioritised using fast and therefore suboptimal methods. This is implemented in our theory by a motive manager. In principle a variety of algorithms could be used for motive management and goal (intention) selection. Our claim is that curiosity driven knowledge gathering must be handled via such a motive management architecture if it is to be done in a way that is efficient when scaling to tens, hundreds, or even thousands of possible learning opportunities.

2.1 Paper summary

The rest of this paper is organised as follows. In the next section we use the George scenario to describe how we bring elements of our theory together to perform curiosity driven learning. In particular the George system shows the use of quantitative and qualitative representations of gaps and uncertainties;

of introspection to detect knowledge gaps; of the motive system to sort and prioritise different learning opportunities; and of the use of learning and unlearning mechanisms. In the section following that we use the Dora scenario to describe how we bring together elements of our theory to perform task driven gap filling. In particular the Dora system shows the use of assumptions and epistemic action effects to fill gaps during task planning; of re-planning with background knowledge to explain negative surprises that induce planning failures; and of the use of open worlds in both these cases. We finish with some concluding remarks. It should be noted that this is a paper giving an overview of the theory. Where the details of a particular mechanism would require too much space we refer the reader back to our relevant published work.

3 George: Representing and Detecting Gaps in a Curiosity Driven System

In the George scenario the main purpose of the cognitive system is to expand and refine its knowledge. This means that (in contrast to the Dora scenario) the system is driven by a generic motivation for detecting and filling the knowledge gaps (rather than this being just a consequence or required step of a given task). To this purpose the system can exploit various situations, which require various degrees of system initiative. Besides the representations of the object properties, like color or shape, the system also maintains measures of confidence in such representations, which allow the system to actively pursue its knowledge expansion goals.

In general there are two main types of behaviors through which the system strives to improve its representation. The first one is based exclusively on the introspection of the existing property models. From a pool of currently maintained property models the robot selects the one that he considers the least adequate (e. g. the most inadequately sampled) and based on that initiates an action that tries to obtain new samples to improve it (e. g. makes a request to the tutor for an object with the properties in question). We call the gap representations generated by this behavior the *introspective gap representations*.

The second behavior type is based on the ability of the system to correctly recognize the properties of the perceived objects. The information the system provides, when analyzing object properties, is twofold: beside the recognition result itself and its confidence, the system also estimates the benefit of the (reliable) additional information about the object properties for the property models. Based on these estimates the system decides whether, in what order and (combined with the recognition result confidence) how to pursue reliable information about the perceived object properties. We call the gap representations of this type the *extrospective gap representations*.

Let me emphasize here one of the major differences between the two behaviors described above that might not be immediately apparent. The former is inherently non-situated, since it completely disregards the current perceptions. Instead it analyzes each internal property model as a whole and establishes how badly does it need new samples to improve it. In contrast, the latter behavior estimates how a perceived sample benefits an internal model. If the former behavior searches for gaps among several property models, where the resulting gap is represented by a whole model (e. g. an under-sampled model), the latter tries to find gaps withing the model itself (e. g. an area that the system believes that is part of the model, but is badly sampled or on model's border, hence often contested by another model). Thus, on the one hand we can have a case of an under-sampled model, which can not benefit much from a perceived object, because its sample falls to a relatively well sampled area of the model, and on the other hand an object whose property fall to a poorly sampled or uncertain area of a well sampled model.

The George system can be roughly divided in the modal and the cross-modal part. The subsystems that make up the modal part (e.g. vision) are typically in charge of processing one type of sensory information (or controlling one piece of hardware equipment). The final result of the modal processing are uni-modal representations of the robot's environment. Uni-modal representations are translated to private beliefs, which are input to the cross-modal cognitive layer. Here the beliefs can be compared, associated and merged into new beliefs to form the a-modal representation of the system environment. Gaps detected on the modal level (modal gaps) are related to the ability of interpretation of the sensory signals (while on the gaps potentially detected on the cross-modal level would be related to the ability of of association between different modal perceptions). Usually, the modal gaps can not be resolved or acted upon within the modality, therefore their representation have to transcend the modal level. In george, the situated gap representations are simply attached to the beliefs about related objects, while the introspective gaps have a special representation on their own on the cross-modal level. In this way knowledge gap information is propagated to the a-modal level where it can be used by motivation and planning subsystems to generate appropriate behavior. In fact in George scenario it is the presence and type of the knowledge gaps that, besides the tasks given by the tutor, motivate the robot behavior, which can be summarized by the three general motivation tiers as follows: (i) attend the tutor's requests, (ii) attend the possible extrospective gaps related to the current situation and (iii) attend the introspective gaps in the current models. The priority of the generated specific motives descends in the same order.

This behavior is exemplified in the section 3.3. The example illustrates, through tutor-robot dialogue, how the tutor's actions motivate robot reactions. We will see how the tutor's specific questions about the objects in the scene trigger the actions of the upper motivation tier — the robot tries to answer the questions. If all the tutor requests are satisfied, the motives of the second motivation tier emerge, generating planning goals that result in robot questions to the tutor. With these questions the robot tries to fill the extrospective gaps related to the objects in the scene. When all the extrospective gaps in the current beliefs are attended and there are no tutor's tasks, it is time to fill the introspective gap in the current property models. Consequently the robot requests from the tutor an object with specific properties. At any time a new request from the tutor or a new object on the scene would make the system reconsider the current motives, surfacing the possible new higher priority motives and unsurfacing the ones with the lower priority.

3.1 Visual Gap Representations

This section presents details about the specific gaps the George is able to detect and represent within its vision subarchitecture.

3.1.1 Representations for visual concepts

The visual concepts are represented as generative models, probability density functions) over the feature space, and are constructed in online fashion from new observations. In particular, we apply the online discriminative Kernel Density Estimator (odKDE) [3] to construct these models. The odKDE 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 odKDE is that is allows adaptation from the positive as well as negative examples [4]. The continuous learning proceeds by extracting visual data in a form of a highdimensional features (e.g., multiple 1D features relating to shape, texture, color and intensity of the observed object) and odKDE is used to estimate the in this high-dimensional feature space. However, concepts such as *color red* reside only within lower dimensional subspace spanned only by features that relate to color (and not texture or shape). Therefore, during online learning, this subspace has to be identified to provide best performance. This is achieved by determining for a set of mutually exclusive concepts (e.g., colors green, blue, orange, etc.) the subspace which minimizes the overlap of the corresponding distributions. The overlap between the distributions is measured using the Hellinger distance as described in [5]. Therefore, during online operation, a multivariate generative model is continually maintained for each of the visual concepts and for mutually exclusive sets of concepts the feature subspace is continually being determined. The set of mutually exclusive concepts can then be used to construct a Bayesian classifier in the recognition phase, when the robot is generating a description of a particular object in terms of its color, shape, etc. However, since the system is operating in an online manner, the closed-world assumption can not be assumed; at every step the system should take into account also the probability of the "unknown model" as described in the following.

3.1.2 Accounting for unknown model

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

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

$$p(z|s) = p(z|m_k, s)p(m_k|s) + p(z|m_u, s)p(m_u|s). \tag{1}$$

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

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

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

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

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

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

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

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

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

In our implementations, the distribution of each i-th alternative of the known model $p(z|M_i, m_k, s)$ is continuously updated by the odKDE [3], 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:

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

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

3.1.3 Detection of knowledge gaps

With above definitions, the knowledge model is completely defined and allows discovery of knowledge gaps. They can be discovered through inspection of the probability distributions. As already mentioned above, we can distinguish two general cases; situated and non-situated knowledge gap discovery.

Extrospective knowledge gap detection In this case, the knowledge gap detection is related to a particular object in the scene. Let us suppose, that we have calculated the AP probability of all the models, including the unknown model, for classification of the observed observation z (4). Let us denote with M_maxap the class with the maximum a posteriori probability, therefore

$$M_\mathrm{maxap} = \arg\max_{M_i}\{p(M_i|\mathbf{z}, s)\}, i = 1 \dots k. \tag{6}$$

and analogously let us denote with M_{maxap2} the class with the second largest a posteriori probability. Now we can detect the gap and the uncertainty in the knowledge as follows:

- **Knowledge gap**:

  $$p(M_{\mathrm{maxap}}|\mathbf{z}, s) << p(M_0|\mathbf{z}, s).$$

  The response of all learned models is very low, the "unknown model" clearly wins. Since the observation can not be modelled by any previously learned model this indicates that it should probably be modelled with a new model.

- **Model uncertainty**:

  $$p(M_{\mathrm{maxap}}|\mathbf{z}, s) \sim p(M_0|\mathbf{z}, s) \bigvee p(M_{\mathrm{maxap}}|\mathbf{z}, s) \sim p(M_{\mathrm{maxap2}}|\mathbf{z}, s)$$

  In this case at least one of the learned models responds quite well, however its response is quite similar to the response of the unknown model or to the model with the second best response. In the first case the observation should probably be classified in the class M_{maxap}, however, since the corresponding model is **weak**, this classification can not be performed reliably. In the second case the models can not distinguish between two classes reliably, which indicates that they are **ambiguous**.

Introspective knowledge gap detection In this case, the knowledge gap detection is not related to a particular object in the scene and it is determined by the inspection of the current models. We formulate the introspection by estimating the expected uncertainty of each concept model in light if all remaining models. In particular, we estimate the uncertainty numerically by sampling a number of samples from a concept model and calculate a measure of classification uncertainty for each sample. The model uncertainty is calculated as the average uncertainty over all the samples. For each sample, the classification uncertainty is estimated as the entropy calculated over the a posteriori probability distribution over all concept models. Formally, the classification uncertainty $\varepsilon(M_i)$ of the i-th concept model, M_i, is calculated as follows

$$\varepsilon(M_i) = \frac{1}{N} \sum_{n=1}^{N} H(\mathbf{z}_n), \tag{7}$$

where $\{\mathbf{z}_n\}_{n=1:N}$ is the set of points sampled from the concept model, i.e., $\mathbf{z}_n \sim p(\mathbf{z}|M_i)$, and $H(\mathbf{z}_n)$ is the Bayes entropy calculated from the posterior distribution over the alternative models $p(M_i|\mathbf{z})$,

$$H(\mathbf{z}_n) = 1 - \sum_{i=1}^{k} p(M_i|\mathbf{z}_n, s). \tag{8}$$

Note that taking the average uncertainty (7) is only one way of providing a measure of model uncertainty. For example, we also might measure the model uncertainty by the maximum over all sample entropies, i.e.,

$$\varepsilon_2(M_i) = \max(\{H(\mathbf{z}_n)\}_{n=1:N}). \tag{9}$$

However, there is a significant difference in the information content that $\varepsilon(M_i)$ delivers in comparison to $\varepsilon_2(M_i)$. While $\varepsilon_2(M_i)$ will in fact determine the model that contains the largest gap, the robot will not be able to communicate this particular gap beyond asking the user to provide a sample corresponding to some concept. This presents a drawback in situations when the sample that the robot internally sampled comes from a low probability region of the feature space. Without any other prior information, the user will mostly provide more or less typical examples of the concepts and these will not likely come from the low-probability feature space. This makes the probability that the user will present a sample, similar to the one that the robot internally generated, quite low. Therefore a much better strategy is to aim at filling a gap in the model whose average uncertainty is highest (7), since there is a higher probability that the tutor-provided sample will correspond to an internally generated sample with a large uncertainty. This is the main reason for a practical preference of expectation-based uncertainties such as (7) over the maximum-based uncertainties such as (9).

3.1.4 Illustrative example

For a better visualization of the knowledge update and gap discovery we will restrict our example to a one-dimensional case. Fig. 1 illustrates detection and filling of knowledge gaps for three cases (feature values) denoted by the circle, the diamond, and the square. The plots in the left column depict the models and the recognition at a particular step in the learning process, while the right column depicts the situation after the system has updated these models considering the detected knowledge gaps and the answers from the tutor.

Let us assume that the circle in Fig. 1 represents the yellow object and that the yellow colour has not been presented to the robot before. Therefore, the corresponding model for colour yellow has not yet been learned and the feature value obtained from the segmented yellow object fails in a not yet modeled area. This value is thus best explained by the "unknown model", which has the highest a posteriori probability. The robot detects this gap in his knowledge and asks the tutor "*Which colour is this object?*", and after

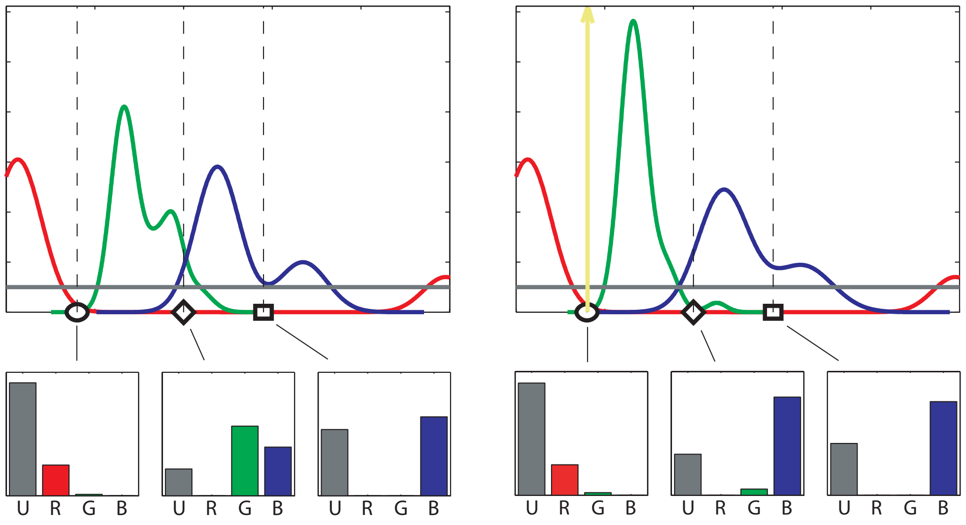

Figure 1: Example of detecting the knowledge gaps and updating the 1D KDE representations. Top row: probability distributions for three colours (red, green, blue lines) and unknown model (gray line) in 1D feature space. Bottom row: a posteriori probabilities for the unknown model (U) and three colours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

the tutor provides the requested information, the robot initializes a model for yellow colour. However, since only one sample does not suffice to build a reliable representation, the yellow colour will only be able to be recognized after some additional yellow objects are observed.

The feature value denoted by a diamond in Fig. 1 is best explained by a green model, however this recognition is not very reliable, therefore the robot asks the tutor: "*Is this object green?*" to verify its belief. After the tutor replies "*No. It is blue.*", the robot first unlearns the representation of green and updates the representation of blue. The corrected representations, depicted in thes in the right column in Fig. 1, then enable the correct recognition as indicated by the second bar plot in the right column of the Fig. 1.

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.

3.2 From Gaps to Motives

In George, the Motivation SA is responsible for transforming the knowledge gaps presented in belief structures into goals for the Planning SA, and man-

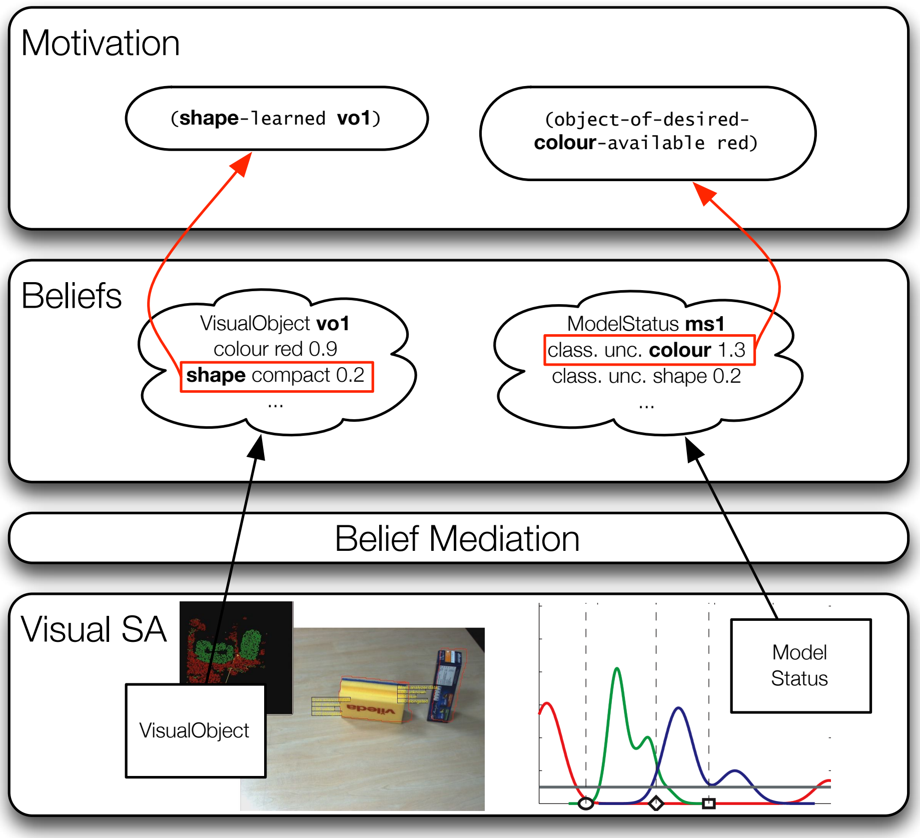

Figure 2: An illustration of information flow for gap representation, detection and goal generation.

aging which goals are pursued when. As this approach works on the belief level our approach requires that the gap-representing structures described in the previous sections are mediated correctly to the belief level. As illustrated in Figure 2, the Motivation SA listens for changes to the belief types produced by these processes. On receiving a change it retrieves the beliefs and inspects them for predefined conditions. For belief features (e.g. colour, shape, identity) it checks the VisualObject belief for the presence of these features. If a feature is missing, or is present with a confidence below a predetermined threshold, this results in a gap being detected. Gaps of this type are directly transformed into planning goals by the Motivation SA by instantiating the template ($\langle feature \rangle$-learned $\langle belief \rangle$). For example, a belief with id *vo1* missing a shape feature would yield a goal of the form *(shape-learned vo1).* For model introspection, VisualConceptModelStatus beliefs are inspected for the information gains described above. If this gain is above a predetermined threshold, then a goal is generated by instantiating the template (object-of-desired-$\langle feature \rangle$-available $\langle value \rangle$). For example, a gap related to the robot's model of white would produce the goal (object-of-desired-colour-available white).

In all cases a motive structure is created to contain the resulting goal which can be annotated with additional management information including the importance of the goal, and the expected planning and execution times for it. All of these motive structures are entered into the lowest level of the motivation management framework (as described in **DR 1.4**), where they are initially *unsurfaced* (i.e. not considered for management, planning or execution). In George we use a drive hierarchy to decide which of the many types of motives should be addressed by the system. This hierarchy can be roughly characterised by three priority levels (lower number means higher priority):

1. Respond to the human.
2. Fill extrospective gaps in knowledge.
3. Fill introspective gaps in knowledge.

Each of these levels is explained in more detail in **DR 7.5**, but in summary level 1 involves responding to questions and learning instructions, level 2 involves filling gaps in beliefs about the objects that are currently visible or part of the situation, and level 3 involves filling gaps in beliefs about the robot's own models. Motives to fill gaps in beliefs about currently visible objects (Scenario 1) are part of level 2. Motives to improve George's internal models by viewing more examples are part of level 3.

Motive activation (i.e. selection for planning and execution) is governed by these levels: if a motive of a higher priority level is present in the system, all lower level motives are suppressed. This allows the robot to be responsive to human interactions, but also allows it to act autonomously to fill gaps when no interaction is taking place. When multiple motives are present at the same level (only possible at levels 2 and 3), all motives are passed to the planner which then produces a plan to achieve a subset of them which maximise knowledge gain whilst minimising cost. Costs are incurred by moving the robot (the head and arm) and by asking questions. Knowledge gain is encoded by an importance associated with each motive by its generator. For the belief features, importance is derived from the confidence of the recognised feature value (a low confidence yields a high importance). For the model introspection, importance is derived from the information gain for each feature. This approach allows principled trade-offs to be made between motives within a level (where important values are directly comparable), whilst inter-level suppression generates appropriate system-level behaviour with respect to interaction.

3.3 An example scenario for extrospective and introspective gaps

The George scenario assumes a robot and a human tutor by a desktop, where the human manipulates with various objects and makes conversation to robot about the situation on the desktop. The robot is equipped with a pan-tilt unit with two sets of visual sensors mounted: (i) the sensors for coarse vision (kinect) have a wider field of view and are used for object detection, (ii) the sensors for precise vision sensors have narrower field of view, but better resolution. The precise vision is used for the object analysis that can recognize the object color and shape.

`The tutor places an object on the desktop.`

The quantitative visual layer (coarse vision with wider field of view) detects a new space of interest. Its representation is propagated as belief to the a-modal level, where it is used to update the planning status. To satisfy its goals, which are currently generated by the extrospective motivation tier, the planner issues actions to move the pan-tilt device with the cameras towards the space of interest and to analyze the object in the center of the view. The visual subsystem, which can now view the object with its precise vision, tries to recognize its shape and color. While it completely fails to recognize the color, the shape recognition is somewhat unreliable. The qualitative analysis of the object also results in the measures of benefit the external (tutor's) information could have for the color and shape models. This information is then, as part of the object belief, propagated to the motivation and planner. Continuing to satisfy its curiosity goals, generated by the extrospective motivation tier, the planner considers the benefits of obtaining additional information about the shape and color and based on that makes a plan related to that object. Since the benefit of obtaining color information or the referred object is greater the plan opts for a question about the color first. Since the color recognition result is completely unreliable the planner decides to ask the tutor an open question.

`Robot:‘‘What color is this object?’’`

`Tutor:‘‘Is is red.’’`

The dialogue subsystem analyzes the tutor’s answer. The information the answer contains is split in two parts: the restrictive part (“it”) is used to identify the referent object, the assertive part (“red”) carries new information about the object. Based on the restrictive information the system resolves the reference to the belief about the object on the desktop. In general, after a successful reference resolution the restrictive information is merged with the information in the object belief, while the assertive information is temporarily kept separate. The dialogue issues an intention to use the color information for learning, which results in learning action in the visual subsystem that improves the color model for red. After the successful learning action, the assertive information is considered verified and is consequently merged in the object belief. The introspective gap representations are updated, too.

`Robot:‘‘OK.’’`

The planner now moves to the next step of the plan. This time the recognition result more reliable (but still not reliable enough) and biased towards the compactness. Hence the planner decides to use a polar question.

`Robot:‘‘Is it also compact?’’`

`Tutor:‘‘Yes.’’`

The shape information is used for a learning action in a very similar fashion as the color information before. This ends the plan related to the object.

`The tutor puts another object on the desktop.`

`Tutor: ‘‘What color is the elongated object?’’`

The system detects another space of interest, which triggers a very similar process as was the case for the first object. The analysis results in a belief about the second object where both color and shape are reliably recognized.

The tutor’s request is analyzed by the dialogue subsystem. The dialogue subsystem uses the restrictive information (“elongated”) to trigger the reference resolution. Since the shape is correctly and reliably recognized, the reference resolution succeeds. This results in an intention about tutor’s request pointing to the object belief. The motivation acts upon this intention and generates a first tier motive, which makes the extrospective motives unsurface. The resulting planning goal makes the robot attend the tutor's request.

```
Robot: ``It is blue.''
```

```
Tutor: ``Correct.''
```

As soon as the action attending tutor's request is completed, the extrospective motives resurface. Since the color and the shape are both correctly and reliably recognized, the planer, based on extrospective gap representations in the object belief, decides not to pursue any action. All the goals generated by the extrospective tier succeed.

After an amount of time of inactivity, in absence of extrospective motives, the motive for introspective behavior surfaces. This makes the planner consider the introspective gap representation in its state. This results in an action that tries to gain new samples for the red color model.

```
Robot:``Show me something red.''
```

3.4 Planning to Fill Gaps

The Planning SA is responsible to generate a plan, i.e., a sequence of actions, that achieves the goal generated by Motivation SA. We can categorize the planning tasks analogously to the priority levels described in Section 3.2 and describe them accordingly in the following.

Respond to the human. Answering questions asked by the tutor is in general no task related to filling gaps, but rather to share the accomplished knowledge. There are, nevertheless, situations where a common ground with respect to reference resolution is missing, and where the planner needs to find a way to fill this gap. The goal is, regardless of the scenerie, always given to the planner as

```
(exists (?v - visualobject)
  (and (is-object-in-question ?v)
       (global-color-question-answered ?v))),
```

where *global-color-question-answered* is the predicate that encodes the kind of question that was asked – in this example, the query had the form

```
Tutor:``What color is...?''
```

The reason we cannot use a simpler kind of goal, where an instance of an object present in the scene is used rather than the existential quantification, is that we might not always have common ground with the tutor on what object we are actually referencing to. For this reason, the reference resolution process marks all objects that are potentially the correct reference with a marker *is-potential-object-in-question*. The planning domain contains an axiom predicate *is-object-in-question* for each visual object, which is set if only a single object is left that is marked as the potential object in question, i.e.

```
(:derived (is-object-in-question ?v - VisualObject)
  (and
    (is-potential-object-in-question ?v)
    (not (exists (?v1 - VisualObject)
         (and (not (= ?v ?v1))
              (is-potential-object-in-question ?v1)))))
)
```

Given the action to answer a global color question,

```
(:action answer-global-color-question
   :parameters (?v - VisualObject ?c - ColorName)
   :precondition
   (and
     (is-object-in-question ?v)
     (= (color ?v) ?c)
   )
   :effect
   (and
     (global-color-question-answered ?v)
     ...
   )
)
```

this leads to the desired behaviour: If there is only one object left that is marked, the axiom will enable *is-object-in-question* of that object, and the precondition to answer a global color question is thus fullfilled (for the correct color only). Executing that action then sets the *global-color-question-answered* predicate used in the goal string to true, and a plan can thereby be found. If there is no common ground between the tutor and George established yet, things are a bit trickier, and we need to additionally consider the actions that

- pointing to an object,
- verify if the object currently pointed at is the object the tutor referred to, and

- a set of actions to verify if an object decsribed by some unique property if the one the tutor referred to.

With these kinds of actions, George is able to establish common ground with the tutor regarding the referred object. It will choose to describe the object if there is a property that only applies to one of the potential objects, as describing is considered the cheaper way. In a case where, for instance, the tutor asks about some objects shape, the robot might decide to describe an object by its color using the action *verify-reference-by-describing-its-color*, which will, for example, result in the question

`Robot:‘‘Do you mean the red one?’’`

In PDDL, this action is modelled as:

```
(:action verify-reference-by-describing-its-color
  :parameters (?v - VisualObject ?c - ColorName)
  :precondition
  (and
    (= (color ?v) ?c)
    (forall (?v1 - VisualObject)
      (or
        (= ?v ?v1)
        (not (is-potential-object-in-question ?v1))
        (not (= ?c (color ?v1)))))
    ...)
    :effect
      (and
        (not (is-potential-object-in-question ?v))
        (increase (total-cost) 1)
      )
  )
```

Alternatively, if there is no way to distinguish one object from all others verbally, George might to choose to point to an object (setting the object fluent *currently-points-at* to the corresponding visual object), and then use the action *verify-reference*:

```
(:action verify-reference
  :parameters (?v - VisualObject)
  :precondition (= (currently-points-at) ?v)
  :effect
  (and
    (not (is-potential-object-in-question ?v))
    (increase (total-cost) 2)
  )
)
```

Fill extrospective gaps in knowledge. The second kind of task refers to the filling of gaps. The first kind of gaps are unresolved ProtoObjects, which are created from SOIs. These are resolved by turning the head to the SOI and starting to analyze the proto object. The planner therefore uses a goal that requires that each proto object has a visual object assigned. When all proto objects are resolved in this way, motivation might decide to fill extrospective gaps on these objects. Such knowledge gaps can only be filled by asking questions to the tutor. If multiple objects are in the scene, the planner tries to describe or point to the object in the same way reference resolution was performed in the first kind of task. It will decide to ask polar questions if the probabilitiy distribution on the feature already points strongly to one value, and will use a global kind of question otherwise.

Fill introspective gaps in knowledge. For the sake of completeness, we also describe the planners role when filling introspective gaps in the knowledge, even though it is rather small. In a scenario where Motivation SA decides that Geroge could learn the most from seeing a red object, it uses the goal string

```
(ask-for-an-object-of-color-goal color_red)
```

to query Plannig SA. This goal can be reached by a plan that contains the single action *ask-for-an-object-of-color color_red*.

4 Dora: dealing with gaps in a task driven framework

Dora is a mobile robot, acting in an unexplored, partially observable environment. Dora can perform a variety of tasks that require gathering of information and therefore reasoning about knowledge gaps.

Dora contains several subarchitectures that all provide the planner with knowledge about the world. The spatial subarchitecture provides us with information about the topological structure of the environment, the categorical subarchitecture tries to classify rooms based on observations. From the conceptual and default knowledge architectures, the planner receives information on how room categories and object existence relate to each other.

The environment in which Dora operates is initially largely unknown. Even the structure of the environment (the number of rooms and places as well as adjacency information) may only be known to a small extent in the beginning. Gaps in the *spatial knowledge* are represented in several ways: The possibility of encountering empty space is represented as special *placeholder nodes* in addition to the places that represent known space. The possibility of finding a new room of a certain type is represented as a

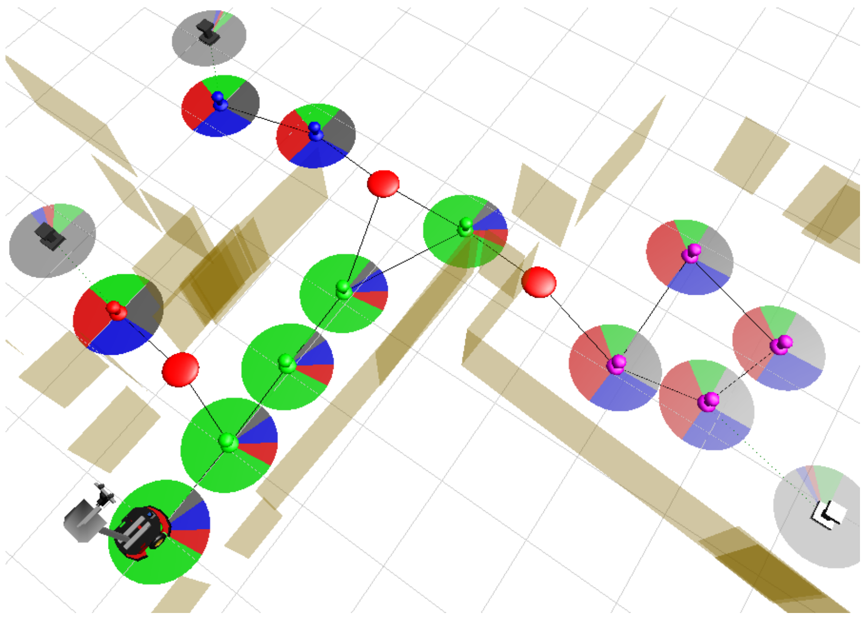

Figure 3: Space in the Dora system is represented by a topological map. Placeholders (in grey) represent unexplored regions, coloured disks show the probability distributions of room categories.

probability distribution attached to each placeholder. The same is true for room categories.

Object co-occurrences are by default provided by the default subarchitecture as $\langle \textit{room category}, \textit{object type}, \textit{probability} \rangle$ triples, specifying the likelihood of finding a certain object in a room of a given type. Once Dora starts to search for that object in a specific *instance* of a room, this prior and the observations must be integrated into a posterior distribution. The conceptual subarchitecture provides the planner with these posteriors for rooms in which object search was started.

In order to achieve either goal-directed or curiosity-driven behaviour in the presence of knowledge gaps, we need planning mechanisms that can reason about these gaps. This means that the planner needs to be able to *identify* gaps in the system's knowledge as well as *reason* about the effects that actions have on the gaps.

A well known approach to this is found in probabilistic planning: The problem is modelled as a partially observable Markov Decision Process (POMDP) and planning is performed directly on this representation. In this model, gaps are modelled by providing a probabilistic *belief state* and effects on gaps are given by *observation models*, that describe what the robot will perceive if the world has a certain state. For our system this approach has two drawbacks: Firstly, reasoning about POMDPs is computationally much harder than classical planning. Especially when the environment is large, solving POMDPs even approximately quickly becomes unfeasible. Secondly, it requires us to assign a probability value to each possible worlds. This is a problem if the robot starts in an unexplored environment (as we would need to model the distribution of all possible environments) and if computing some of these distributions is an expensive operation on its own (such as determining the most likely locations for some object inside a room).

This is why we use a continual planning approach: the planner creates a plan that will lead to the goal for some possible world, and replanning if that plan fails at some point. In our model, gaps are implicitly modelled by *not* assigning values to variables and changes in knowledge are tracked via additional predicates. We use a special kind of epistemic action called *assumption* to model which worlds are possible and to distinguish between likely and unlikely possibilities.

4.1 Object fluents and knowledge level predicates

The representation of classical STRIPS-like planning problems employs the *closed world assumption*: The planning state is specified as a set of ground atoms and all atoms that are not known to be true are assumed to be false. PDDL 3.2 introduced the concept of *object fluents*. Object fluents represent multi state variables and by default, every object fluent has a undefined value. Thus, knowledge gaps are encoded implicitly: Any object fluent that is not defined in the system's initial planning state is a potential gap in the robot's knowledge. This allows us to distinguish between knowing whether a fluents has a value v and not knowing the value of a fluent, but no other epistemic states. In particular, we cannot reason about changes in knowledge *without specifying exactly what we will know* (i.e. the value v).

For this reason, we use *modal predicates* to explicitly represent the fact that the planner knows the value of a given fluent without having to actually set the fluent. For an object fluent X, the *knowledge predicate* $\mathcal{K}X$ is true whenever it is explicitly set to true by an action or if X has a value different from undefined. A second modal predicate is the *assumption predicate* $\mathcal{A}X = v$. It is true when the planner assumes that fluent X has value v. All assumption predicates for a specific fluent X are mutually exclusive, and if X has a value v, $\mathcal{A}X = v$ holds. These two sets of modal predicates allow us to model four separate knowledge states:

- X is undefined an no modal predicates about X are true: We make no assumptions about X and do not know its value.
- $\mathcal{K}X$: We know that we know the value of X, without specifying that value.

- $\mathcal{A}X = v$: We assume a value for X, without knowing whether it is true.
- $\mathcal{A}X = v \wedge \mathcal{K}X$: We know X has the value v, which is equivalent to the assignment $X = v$. In particular, $X = v \Rightarrow \mathcal{A}X = v \wedge \mathcal{K}X$ holds for all planning states.

The implicit representation of gaps as undefined fluents means that the number of gaps the planner may have to reason about can become very large or that a large number of gaps must be explicitly closed in the initial state (everything that is known to be false must be specified in the state description). For example, the connectivity between N places in an environment as shown in Figure 4 can be encoded by a N^2 `connected` fluents. As the connectivity graph is planar, this would require us to specify $O(N^2)$ false `connected` fluents. It turns out that this can usually be avoided if those "false" gaps do not influence the planning process. For example, we could restrict the planner to reason only about gaps in connectivity knowledge between close places.

4.2 Assumptions and epistemic actions

Assumptions in our planning system have several uses. They

- restrict the planner to reason about a set of "sensible" gaps,
- allow us to reason in a limited way about probabilistic states while using a classical planner [2], and
- are a way to compactly represent the large probability distributions that occur when planning in uncertain, open worlds.

An assumptive action has one or more effects on assumption predicates and may depend on zero or more assumptions. Additionally, once an assumption for a fluent X, $\mathcal{A}X = v$ has been made, no other assumption that has $\mathcal{A}X = v', v \neq v'$ as an effect is allowed anymore, enforcing the mutex condition on the assumption predicates. So once we made the assumption that a room is a kitchen ($\mathcal{A}$`(category room0)` $=$ `kitchen`), we cannot, in the same plan, make an assumption that it is an office. Assumptive actions are purely virtual for the planner and are not executed by the rest of the system. However, they can have *costs* in the sense that some assumptions may be more likely than others. These costs are taken into account by the planner and it will try to find plans that contain only likely (i.e. cheap) assumptions while keeping the plan duration short.

Assumptions are mainly derived from two sources: if an object feature X has a probability distribution over a number of features, $P(X = v_i) = p_i$, we create an assumption for each v_i. For example, if a room has a probability of

0.4 of being a kitchen this would be represented by a *grounded* assumption like the following:

```
(:action assume-category-room0-kitchen
   :effect (A (category room0) kitchen)
   :probability (0.4))
```

Executing this assumption at the start of a plan will set $\mathcal{A}$`(category room0)` $=$ `kitchen` to true.

The second type of assumptions is those that are explicitly defined as part of the planning domain. They allow the domain designer to specify relations between facts [1]. The co-occurrence between object types and room categories (e.g. that cereals are likely to be found in kitchens) is modelled in Dora as follows:

```
(:action assume-default-object-in-room
   :parameters   (?l - label ?r - room ?c - category)
   :precondition (and (A (category ?r) ?c)
                      (not (K (p-obj-in-room ?l ?r))))
   :effect       (A (obj-exists ?l in ?r) true)
   :probability  (p-obj-in-category ?l ?c))
```

This assumption uses *default knowledge* which is stored in the numeric fluents `(p-obj-in-category ?l ?c)` to assume the existence of an object of type `?l` in a room, if the assumption that the room is of category `?c` holds, which is ensured by the first line of the precondition. Of interest is also the second precondition: It makes sure that we can only make the default assumption, if we have no more specific knowledge about co-occurrences in this room. If we had started to search the room for an object (say, a cereal box), the conceptual subarchitecture would assign a value to the fact `(p-obj-in-room cereal-box room0)`. As this value is more accurate than the default knowledge (as it includes the observations the robot made), we are no longer allowed to make assumptions using default knowledge. Instead, we can make the assumption directly:

```
(:action object-in-room
   :parameters   (?l - label ?r - room)
   :effect       (A (obj-exists ?l in ?r) true)
   :probability  (p-obj-in-room ?l ?r))
```

Note that we can represent the fact that *we know the probability* of a relation. This allows us to use the more general `default-object-in-room` assumption in absence of more specific knowledge.

Knowledge gathering is modelled by actions that have knowledge effects. These knowledge effects may be the primary effect of an action (i.e. sensing actions) or a secondary effect. The following operator shows how we model that moving to a place will always tell us in which room this place is:

```
(:action move
   :parameters (?from ?to - place)
   :precondition (and (or (connected ?from ?to)
                          (connected ?to ?from))
                      (= (is-in dora) ?from))
   :effect (and (= (is-in dora) ?to)
                (= (placestatus ?to) trueplace)
                (K (in-room ?to))))
```

Sensing actions can be distinguished further into *conditional* and *unconditional* sensing actions. The former depend on an assumption being true to be successful (e.g. object detection will only result in the object being found if it is in the place the robot looks), the latter will always succeed (e.g. asking a human what kind of room the robot is in). The *search-for-object* action is a conditional sensing action:

```
(:action search-for-object
   :parameters (?l - label ?r - room ?p - place ?o - visualobject)
   :precondition (and (= (is-in dora) ?p)
                      (viewcones-exist ?l in ?r)
                      (A (in-room ?p) ?r)
                      (= (label ?o) ?l))
   :effect (when (A (related-to ?o) ?r)
                 (K (related-to ?o))))
```

The effect `(K (related-to ?o)` is only realised if we made the assumption `(A (related-to ?o) ?r)` that the object is actually in that room. We also require the assumption that the position of the robot is in the room we want to search. Had we made this a normal precondition, `(= (in-room ?p) ?r)`, we could never plan for searching in unexplored space, as the `in-room` fluent is only set upon moving to a place.

4.3 An example scenario for task-driven information gathering

We illustrate how the mechanisms we previously described contribute to Dora solving an object search task in an unknown environment. The robot is given a task of finding a specific object in an initially unexplored environment. This translates to a goal formula in PDDL: `(K (related-to visualobject0))`, which is the epistemic goal of determining the value of `(related-to visualobject0)`.

The resulting plan might look this:

```
(assume-leads-to-room placeholder0 room0 kitchen)
(assume-default-object-in-room cereal-box room0 kitchen)
```

```
(assume-object-location visualobject0 cereal-box room0)
(move dora placeholder0)
(create-viewcones  cereal-box room0 placeholder0)
(search-for-object cereal-box room0 visualobject0 placeholder0)
```

Each of these action either creates or requires an epistemic effect.

The first action creates the assumption that `placeholder0` leads to an until know unknown kitchen which is given the temporary name `room0`. The action makes to assumptions true: That the place *is* in the new room, $\mathcal{A}$`(in-room placeholder0)` = `room0`, and that the new room is a kitchen $\mathcal{A}$`(category room0)` = `kitchen`.

As we have no specific instance knowledge about the likelihood of finding a cereal box in `room0` ($\mathcal{K}$`(p-obj-in-room cereal-box room0)` is false), second action uses the previous assumption that `room0` is a kitchen and default knowledge to make the assumption that there exists cereal-box in that room `(A (obj-exists cereal-box room0) true)`. In order to have a concrete cereal-box object to plan with, the next assumption uses the abstract `obj-exists` assumption an makes the assumption that a new `visualobject0` is in `room0`: $\mathcal{A}$`(related-to visualobject0)` = `room0`.

The `move`-action is the first physical action the planner executes. As such it has a physical effect (Dora will be at the new place) but also the epistemic effect $\mathcal{K}$`(in-room placeholder0)`: After we have moved to a place, we will know for sure in which room it is.

Before Dora can search for an object, we must create a set of *viewcones*, which represent locations and orientations from which we expect most likely see the cereal box if it exists. The `create-viewcones`-action has no immediately obvious epistemic preconditions. Its only condition is that Dora needs to be at a place in the room in which we want to create the viewcones. But similarly to the search action, the condition that `placeholder0` is in `room0` is not part of the planning state at this point, so the `create-viewcones` action depends on the assumption we made that this is true in the first action. Finally, after the viewcones are created, we search for the object. As described above, the action has a conditional knowledge effect: assuming the `visualobject0` is in the room, we will know its location afterwards. As the assumption is satisfied by the `assume-object-location` action, the effect $\mathcal{K}$`(related-to visualobject0)` is realised, satisfying the goal.

Let us now look at the same task after the `move` action has been executed, and an new room has actually been found. If the planner replans at this point, the plan will look as follows:

```
(assume-category-room0-kitchen)
(assume-default-object-in-room cereal-box room0 kitchen)
(assume-object-location visualobject0 cereal-box room0)
(create-viewcones  cereal-box room0 placeholder0)
```

```
(search-for-object cereal-box room0 visualobject0 placeholder0)
```

Now that the planner knows that `room0` exists, the first assumption is replaced by a ground assumption that is created from the category probabilities provided by the categorising process. The other assumptions and the actions after the `move` action remain the same. The `(A (in-room placeholder0) room0)` precondition of the viewcones and search actions is no longer satisfied by the `leads-to-room` assumption, but by the initial state: As `placeholder0` *is* in `room0`, `(in-room placeholder0)` is set to `room0`, which by our definitions implies $\mathcal{A}$`(in-room placeholder0)` = `room0`.

If the planner has to replan during the search action (e.g. because an action fails randomly), the `(p-obj-in-room cereal-box room0)` will have a value from the conceptual subarchitecture. So the planner is no longer allowed to use default knowledge to make assumptions about cereal boxes in `room0`. If the existence probability is still high enough, this will lead to the following plan, where the instance-specific probability is used to make the `obj-exists` assumption:

```
(assume-object-in-room cereal-box room0)
(assume-object-location visualobject0 cereal-box room0)
(search-for-object cereal-box room0 visualobject0 placeholder0)
```

If the plan succeeds and the object is found, `(related-to visualobject0)` will be set to a value and the goal will be reached. Otherwise, the planner will either try to reach the goal in a different way or fail if no other way could be found.

4.4 Explaining surprises using assumptions

Making assumptions explicit can be helpful if we want to reason about the sources of planning failures or other observations that were not predicted by the planner. A more exhaustive description on how we use planning technologies to explain "surprises" can be found in **DR 7.3**, here we want to briefly outline how our representation of gaps facilitates reasoning about unexpected behaviour.

Explaining surprises involves finding a probable set of assumptions, that if they were true, would cause the observations the robot made during the execution of its plan. In our object search example, the task may fail because `search-for-object` actions do not result in the desired effect, $\mathcal{K}$`(related-to visualobject0)`. We are therefore looking for a set of likely assumptions that would cause that behaviour. The explanation for violated assumptions may be quite mundane: If we do not find a cereal box in the kitchen, despite looking everywhere, a possible explanation for that could be that cereals are simply not there.

While finding explanations with a high likelihood is the best case scenario, this is often not the case. If we know, for example, that the probability of cereals being in kitchens is very high, the previous explanation is not very convincing. In cases where we cannot find a likely explanation, we prefer to find those that we have no information at all about – it is preferable to find explanations for which we do not know whether they are likely than to find those we *know* to be unlikely. In our example, we could assume that the cereals are inside another container, therefore being invisible to the robot's camera.

The explanations that are generated by these processes can then be picked up again by the motivation subsystem to generate new goals to try to verify whether the explanations was correct. In our previous examples, this may involve entering into dialogue with a human, asking them if there are any cereals in the room, or trying to detect the assumed container. Especially if the explanations we find are those with unknown likelihood, this allows the robot to use its failures to identify knowledge gaps to fill in the anticipation that the new information may prove to be useful for future tasks.

5 Conclusion

In this paper we have given an overview of how the elements of our theory of how gaps may be identified and filled fit together to constitute a coherent whole. To illustrate this we have organised the paper around two systems that we describe in some detail. These are the George robot system which performs curiosity driven gap detection and learning; and the Dora robot system, which performs task driven gap filling and explanation. We described how in a curiosity driven system the robot raises motives to learn once gaps are detected, and these are prioritised via a motive management system. This also allows us to set non-learning motives as having priorities. In the George system for example questions from the human generate the highest priority motives. In Dora there is only ever one motive, which is to perform the task set by the human operator. But it is clearly the case that such systems can be trivially combined using the motive management system by setting the priorities of the tasks set by the human (for Dora) along with the questions (asked of George) and the learning motives that are set by the robot (George). Such a system would be inefficient in that it would not intelligently interleave curiosity driven and task driven activity. This final piece of the puzzle we will describe in DR.1.4. In that paper we show how ideas from planning with deadlines can allow the robot to more efficiently interleave task driven to curiosity driven activity.

Acknowledgments

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

References

[1] A. Aydemir, M. Göbelbecker, A. Pronobis, K. Sjöö, and P. Jensfelt. Plan-based object search and exploration using semantic spatial knowledge in the real world. In *Proc. of the European Conference on Mobile Robotics (ECMR 2011)*, Örebro, Sweden, sep 2011.

[2] M. Göbelbecker, C. Gretton, and R. Dearden. A switching planner for combined task and observation planning. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI 2011)*, page NA, 2011.

[3] M. Kristan and A. Leonardis. Online discriminative kernel density estimation. In *International Conference on Pattern Recognition*, pages 581–584, Istanbul, Turkey, 23-26 August 2010.

[4] M. Kristan, D. Skočaj, and A. Leonardis. Online Kernel Density Estimation for interactive learning. *Image and Vision Computing*, 28(7):1106–1116, July 2010.

[5] D. Skočaj, M. Kristan, and A. Leonardis. Continuous learning of simple visual concepts using Incremental Kernel Density Estimation. In *VISSAP 2008*, pages 598–604, 2008.

[6] 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, and Michael Zillich. Self-understanding and self-extension: A systems and representational approach. *IEEE Transactions on Autonomous Mental Development*, 2(4):282 – 303, December 2010.