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

DR 7.3:
Analysis of a Robot that Explains Surprise

Jeremy L. Wyatt[1], Marc Hanheide[2], Andrzej Pronobis[3], Kristoffer Sjöö[3], Alper Aydemir[3], Patric Jensfelt[3], Moritz Göbelbecker[4], Charles Gretton[5], Graham S. Horn[1], Richard Dearden[1], Miroslav Janicek[6], Hendrik Zender[6], and Geert-Jan Kruijff[6]

[1]*BHAM, Birmingham* [2]*Lincoln University (formerly BHAM)*
[3]*KTH, Stockholm* [4] *ALU-FREI, Freiburg*
[5]*NICTA, formerly BHAM* [6] *DFKI GmbH, Saarbrücken*
⟨`jlw@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

In this report we describe our experimental analysis of the the Year 3 Dora system, a robot that could plan to achieve a variety of tasks in an environment with undiscovered objects, rooms etc. In addition this robot was able to explain surprising planning failures. In the second attachment to this deliverable we give more technical detail on the method used to achieve this.

Executive Summary

This report presents an analysis of the period 3-4 Dora system. This system was able to plan information gathering activities necessary to achieve a task given by a human. The robot can reason about a variety of entities, including making assumptions about the world necessary to form a plan, modelling limited open worlds, modelling the epistemic effects of actions, replanning on the fly, and switching between decision theoretic and non-decision theoretic planning as necessary. In addition the system is able to explain surprises that result in planning failures, such as when an expected outcome essential to a plan does not occur. It seeks to explain these using the framework of assumptions (and additional background knowledge), just as it uses assumptions to produce plans under incomplete knowledge in the first place. We present a case based experimental analysis of the main system in the first attachment, and present the use of assumptions to produce explanations for surprising planning failures in the second attachment.

Role of explanations and surprises in CogX

In our plans for CogX the ability to explain surprises is one of the final stages of the scheme for self-extension. By explaining surprising results in terms of assumptions and additional background knowledge the robot is able to form hypotheses about the existence of additional objects, or additional relations between objects, or both. These hypotheses can then be tested using additional plans, and then added to the robot's knowledge thus creating a second route to self-extension (the first being exploration based on the initial knowledge and tasks as demonstrated in several of the systems presented in CogX (Dora 1, Dora2, George 1-3, Dexter 2).

Contribution to the CogX scenarios and prototypes

The results presented in this report are an analysis of the Dora system, and are thus related to the scenario on task driven information gathering and self-extension.

1 Tasks, objectives, results

1.1 Planned work

The task to which this work contributes is Task 7.6 (*Experimental study of explanation with limited extension*). The aim was to analyse a robot that can extend its representations in a limited way (e.g. extending its map). The objective addressed in the work is Objective 11 (*A robotic implementation of our theory able to complete a task involving mobility, interaction and manipulation. In the face of novelty, uncertainty, partial task specification and incomplete knowledge*). In this deliverable we have focussed on analysing a robot implementation concerned with mobility and interaction with a human, where the knowledge of the robot is incomplete at the start of the task.

1.2 Actual work performed

The attachments to the deliverable describe the results of the work performed during year 3 and 4 on the Dora system to enable it to explore and fill knowledge gaps in a task driven manner, and to explain surprises that result in planning failures when they occur. The Task 7.6 has been addressed here by a case based experimental analysis of the robot's ability to perform a variety of tasks in an environment under incomplete knowledge, by the ability of the robot to extend its representations, by the ability of the robot to plan to do so, and also by the ability of the robot to explain surprising planning failures. The objective O11 has been addressed by the development of this same system, and by the integration in the robot of various elements of our theory.

1.3 Relation to the state-of-the-art

The major difference between Dora and other previous robot systems that perform similar tasks (object search, autonomous mapping, room categorisation) is that i) Dora is re-taskable across a range of tasks whereas previous systems typically perform just one task, ii) Dora reasons about open worldness, and iii) employs a switching planner that enables both satisficing and optimising style planning.

With regard to retaskability, there are many instances of systems that perform active SLAM [4, 6] by using path planners in continuous or quantised state spaces that explicitly plan information gain over many steps, but only for that specific task. Similarly, in object search there are a number of approaches that plan within an information space, expressing the value of particular viewing locations by modelling both sensor behaviour and prior belief about object location [8]. In these planning is often greedy one step lookahead for view selection [1, 5], although [7] reasons about information over multiple steps.

All these approaches, however, path plan rather than perform task level planning, and do so within an essentially closed world using probabilistic representations of state uncertainty. Most other probabilistic planners or path planners for robots employ unstructured representations of state ([7] is an exception) that make path planning or task specific planning easy, but which do not easily lend themselves either to re-taskability or to planning in open worlds. It is a difficult problem to extend probabilistic approaches to reasoning about open worlds (i.e. where new objects, rooms etc may appear). We have developed instead developed extensions to our continual planning approach that allow planning in limited open worlds, and thus enable the robot to reason about the benefit of activities such as searching for a room of the type that is likely to contain the object searched for [2]. In addition we employ a switching planner, that swaps between a classical and a decision theoretic representation of the planning domain. This has allowed us to produce plans that reason about trade-offs quantitatively when this is computationally feasible, and the remainder of the time produce satisficing plans. Other approaches (of which we are aware) to task planning for robots that are retaskable only produce satisficing plans e.g. [3].

References

[1] A. Andreopoulos, S. Hasler, H. Wersing, H. Janssen, J.K. Tsotsos, and E. Korner. Active 3d object localization using a humanoid robot. *Robotics, IEEE Transactions on*, (99):1–18, 2011.

[2] 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. *Proc. of the European Conference on Mobile Robotics (ECMR 2011), Orebro, Sweden*, 2011.

[3] C. Galindo, J.A. Fernández-Madrigal, J. González, and A. Saffiotti. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955–966, 2008.

[4] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103, 2009.

[5] K. Shubina and J.K. Tsotsos. Visual search for an object in a 3d environment using a mobile robot. *Computer Vision and Image Understanding*, 114(5):535–547, 2010.

[6] R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 661–666. IEEE, 2005.

[7] J. Velez, G. Hemann, A. Huang, I. Posner, and N. Roy. Planning to perceive: Exploiting mobility for robust object detection. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Freiburg, Germany, 2011.

[8] Y. Ye and J.K. Tsotsos. Sensor planning for 3d object search. *Computer Vision and Image Understanding*, 73(2):145–168, 1999.

2 Annexes

2.1 Reasoning about Epistemic Actions and Uncertainty for Autonomous Knowledge Gathering

Bibliography M. Hanheide, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, M. Göbelbecker, C. Gretton, G.S. Horn, R. Dearden, M.Janicek, H.Zender, G.J. Kruijff, J.L. Wyatt“Reasoning about Epistemic Actions and Uncertainty for Autonomous Knowledge Gathering”. To be submitted.

Abstract In any real world task a robot tries to accomplish, it faces two significant challenges that it needs to deal with: (i) its knowledge about the world is incomplete, so substantial knowledge required to successfully achieve a given goal is missing; and (ii) the knowledge it might have about the world is uncertain, due to noise in sensing and/or unreliability of accessible knowledge sources. Nevertheless, we expect our robot to behave intelligently to achieve the given goal robustly and efficiently. In this paper we present a robot system and its architectural underpinning that addresses these challenges. It can accomplish a variety of different epistemic goals in order to extend its knowledge about the world and it features a probabilistic approach to representation, reasoning and planning that allows it to generate goal-driven behaviour in a world full of uncertainties. The robot features a novel *switching planner* that allows the system to schedule actions implemented by a number of *competences* that gather knowledge from various knowledge sources, such as: interactively by asking humans; from sensing the world through the robot’s own sensors; or by exploiting common-sense background knowledge gathered from web resources. We demonstrate and analyse the behaviour of our system in real world runs with three different goals given to the robot: To autonomously explore an unknown map; to learn about the category of rooms in this map (e.g. kitchen, corridor, etc.); and to autonomously determine the location of a specific object. We show that the robot autonomously invokes competences that yield the intended information gain in order to accomplish each task.

Relation to WP WP7 is about the demonstration of the ideas in CogX as a complete systems level theory that works in real robot systems. The work presented in this deliverable is a description of the Dora system that brings together numerous contributions from across CogX. WP7 also concerns the analysis of the robot systems we develop. This deliverable also presents an case based analysis of the abilities of the Dora system and a detailed presentation of the methods used to performance explanations of surprising planning failures.

2.2 Explaining Execution Failures in Continual Planning

Bibliography M. Göbelbecker "Explaining Execution Failures in Continual Planning". Technical Report.

Abstract Continual planning is an effective approach to decision making in uncertain dynamic worlds. It involves creating plans based on assumptions about the real world and replanning if those plans fail. We discuss methods for making these assumptions explicit and providing explanations why a continual planning task may have failed or produced unexpected outcomes.

Relation to WP In this WP we developed the Dora system, which is able to explain surprising planning failures. This attachment explains how that explanation mechanism is implemented in terms of assumptions.

Reasoning about Epistemic Actions and Uncertainty for Autonomous Knowledge Gathering

Marc Hanheide, Andrzej Pronobis, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Moritz Göbelbecker, Charles Gretton, Graham S. Horn, Richard Dearden, Jeremy L. Wyatt, Miroslav Janicek, Hendrik Zender, and Geert-Jan Kruijff

non-public Technical Report

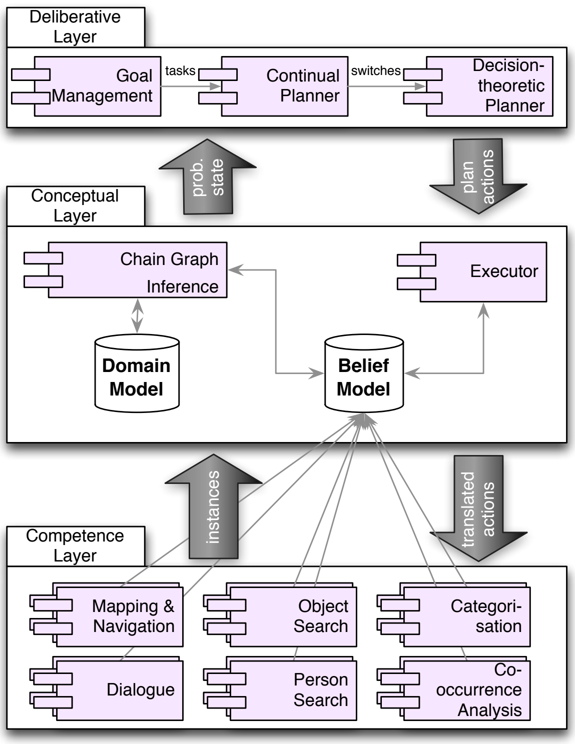
Abstract

In any real world task a robot tries to accomplish, it faces two significant challenges that it needs to deal with: (i) its knowledge about the world is incomplete, so substantial knowledge required to successfully achieve a given goal is missing; and (ii) the knowledge it might have about the world is uncertain, due to noise in sensing and/or unreliability of accessible knowledge sources. Nevertheless, we expect our robot to behave intelligently to achieve the given goal robustly and efficiently. In this paper we present a robot system and its architectural underpinning that addresses these challenges. It can accomplish a variety of different epistemic goals in order to extend its knowledge about the world and it features a probabilistic approach to representation, reasoning and planning that allows it to generate goal-driven behaviour in a world full of uncertainties. The robot features a novel *switching planner* that allows the system to schedule actions implemented by a number of *competences* that gather knowledge from various knowledge sources, such as: interactively by asking humans; from sensing the world through the robot's own sensors; or by exploiting common-sense background knowledge gathered from web resources. We demonstrate and analyse the behaviour of our system in real world runs with three different goals given to the robot: To autonomously explore an unknown map; to learn about the category of rooms in this map (e.g. kitchen, corridor, etc.); and to autonomously determine the location of a specific object. We show that the robot autonomously invokes competences that yield the intended information gain in order to accomplish each task.

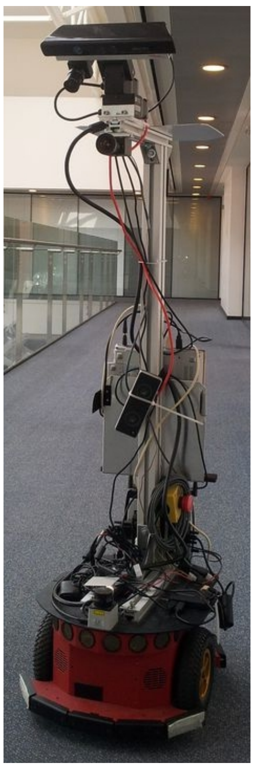
1 Introduction

Autonomous robotic systems have impressively advanced in recent years and they have made significant progress on various individual problems and application scenarios. An overarching challenge for such robots is to gather knowledge about their environment and situation that is required to accomplish given tasks autonomously. Most approaches either feature explicit training or tutoring phases, or have been endowed with pre-defined, specific knowledge. In real world however, robots can and should learn from various knowledge sources to really be adaptive and robust. These could be (i) humans that can explicitly show or tell the robot what it needs to know (e.g. the robot can ask what room it is currently in), (ii) observation of the environment through different sensors (e.g. vision), to relate the robot's perception to pre-determined, conceptual and categorical information yielding the classification of perceived instances in the world (e.g. recognise an object in the robot's field of view using pre-trained object classifiers), or (iii) to access information online querying about specific instances perceived (e.g. querying the visual appearance of an object from an image search engine to compare it to the current visual input). What is common across those different classes of knowledge sources is that the result of these actions is inherently uncertain with respect to the specific situation the robot is confronted with. Humans might or might not have the information the robot asks for; or they might not even be truthful. Observing the environment autonomously and striving to, e.g., detect an object with a camera, is even more prone to errors given the state of the art of general object recognition [1]. Querying the web is indeed an excellent source for generic knowledge, which however may prove entirely wrong or very insignificant in the robot's actual environment and situation. Hence, a robot has to able to reason and deal with the inherent uncertainty of the knowledge it might get from these various sources.

Our approach presented in this paper is to employ a domain-independent probabilistic planning framework that can reason about epistemic actions and choose between them to generate the most rewarding behaviour in terms of epistemic benefit for a variety of tasks. We combine this domain-independent framework with a conceptual probabilistic map accommodating the acquired and pre-determined knowledge about the world to realise a robot system that can autonomously accomplish different epistemic goals in the real world by choosing from three major classes of epistemic actions: (i) Consulting a human verbally, (ii) looking for specific classes of physical objects visually, and (iii) exploring into yet unknown space of the environment. We are able to show that with our approach, our robots can accomplish different goals, that each are usually targeted by dedicated "expert robot systems", and that we achieve comparable performance; and that it can generate efficient and robust goal-driven behaviour by accounting for the uncertain nature of the knowledge gathered explicitly. Hence, the two key contributions of this work are (i) the domain-independent, probabilistic planning framework allowing us to pursue a variety of different goals by literally just the exchange of a string describing an intended goal state, and (ii) the layered architecture of our self-extending mobile domestic robot "Dora", built around a conceptual map and featuring a number of knowledge-gathering state-of-the-art components, ranging from object detection, over adaptive dialogue, to navigation and mapping. The system presented in this paper is the advancement of its predecessor systems [2] which solely focused on efficient and robust object search. Instead, in this paper we demonstrate our approach to be feasible for a variety of different goals and a wider range of actions and behaviours, and we reflect on the architectural aspects of the system. We will see (in our experimental evaluation), that with one and the same system running (continuously) our robot Dora can explore into new rooms, interactively become more confident about the categroy of those rooms, and also find an object. While each of these abilities (exploration and mapping, interactive room learning, and object search) have certainly been addressed individually, this to our knowledge is the first system, that can be tasked to extend its knowledge that broadly in a goal-driven and intelligent way.



(a) Dora's software architecture is an instantiation of the CogX Layered Architecture Schema (CLAS).



(b) "Dora"

Figure 1: Software architecture and hardware setup.

2 System Overview

The Dora software architecture (Fig. 1(a)) is an instantiation of the CogX Layered Architecture Schema (CLAS). CLAS is a progression of our previous work on architectures [3, 4, 5]. It builds on the CoSy Architecture Schema [6] and implemented system instances (like the one presented in this paper) are built using the CAS Toolkit (CAST) [7], a component-based, event-driven integration framework that has been employed in various predecessor systems [8, 9, 2]. In the following, we will explicitly outline where CLAS employs concepts similar to the ones put forward by CAS and where it deviates.

CAS promotes a decomposition for system design into *subarchitectures* that accommodate a number of components which exchange predominantly *modal* information through the modification of contents of *working memories*. Hence, our systems [8, 9, 3, 4, 5] always featured individual subarchitectures for visual processing, dialogue, spatial mapping, etc, that are built around representations of certain modalities; here vision, speech, and maps, respectively. These subarchitectures all generally endowed with equal rights, permissions, and policies. However, already in these previous architectures we identified the special role of some subarchitectures that did not really fall into the category of purely modal subarchitectures. In [6] and [3] we described the *binding* subarchitecture to support amodal, unified representations that facilitate reasoning and planning across different modalities, and also proposed a *planning* subarchitecture that exploits the amodal representation to devise plans to generate intelligent behaviour. CLAS, as illustrated in Fig 1(a), expands on these ideas and makes explicit the *functional* decomposition into the three layers that was implicitly proposed in these earlier architectures. CLAS features *probabilistic* amodal abstraction of both domain knowledge and instance relations in a *conceptual layer*, and a *deliberative layer* that contains advanced planning components which can exploit and handle the probabilistic nature of the belief state represented in the conceptual layer. In particular, the planners can generate plans for epistemic goals. These two layers are complemented by the *competence layer*, which contains the different modal subarchitectures in a CAS system. The conceptual layer can be seen as a mediation layer, facilitating decoupling between the deliberative and competence layers.

Functionally, and with regard to many of the competences involved, [2] provides the foundation for the work we present here. But while in [2] we solely focused on the benefit of exploiting uncertain knowledge and sensing for the task of "object search", in this paper we instead demonstrate our approach to be feasible for a variety of different goals and a wider range of action and behaviours. We shall now describe the three layers in greater detail.

2.1 Competences

At the lowest layer CLAS features *competences*. Competences consist of a set of components designed to jointly provide a certain well encapsulated and independent behaviour or functionality for our system. Generally speaking, a competence is thought of as a subsystem that can *provide knowledge* to the conceptual layer and can possibly be *tasked* to carry out actions. In most cases, a competence is implemented as a subarchitecture in the CAST notion.

Examples relevant to this paper are competences (shown as stacks of components in Fig. 1(a)) to autonomously move the robot about safely ("Mapping & Navigation"), to perceive the environment ("Object Search", "Person Detection", "Categorisation"), to interaction with humans ("Dialogue Management") and to gather knowledge from the web ("Co-occurrence Analysis"). They can be tasked by higher layers to execute their respective behaviours. For instance, the "Mapping and Navigation" competence can be tasked to move to a dedicated place, as will be discussed in Sec. 3.1 in detail. In CLAS, the competence layer is very domain-specific. Competences determine what the robot can do and how it will do it. They implement the different epistemic actions the robot can execute. But how the competences are arbitrated, combined, and sequenced is not decided on this layer. Dora's competences are subject to detailed discussion in Sec. 3, where we will outline how they are realised and what kind of knowledge they will enable the system to gather.

2.2 Deliberative Layer

At the top of CLAS the deliberative layer plays a central role to actually generate goal-driven behaviour. From a planning perspective we are confronted with a number of important but contrary challenges. On the one hand, planning and execution monitoring must be lightweight, robust, and timely. Those processes must seamlessly accommodate exogenous events, changing objectives, and the underlying unpredictability of the environment. On the other hand, in order to act intelligently the robot must perform computationally expensive reasoning about contingencies, and possible revisions of its subjective belief according to quantitatively modelled uncertainty in acting and sensing. Hence, the specific challenge for us here lies in the *uncertainty* of action outcomes and the uncertainty of knowledge available to the system. Despite this uncertainty, the robot must take sensible decisions and devise appropriate plans. The deliberative layer has to invoke the different competences of the system sequentially in order to generate the most *robust* and *efficient* behaviour to gather information. Our approach to these challenges is to switch between decision-theoretic and classical modes of planning at different levels of abstraction. A classical system (part of the "Continual Planner" in Fig. 1(a)) quickly solves a determinisation of the probabilistic problem at hand. Then a decision-theoretic system quickly solves abstract decision problems derived using the classical plan and the probabilistic belief state. Overall, this approach allows the system to generate intelligent behaviour under uncertainty in a timely manner. Most notably, we designed this layer to be domain-independent. The switching planner (to be described in detail in Sec. 5) is a domain-independent state-of-the-art continual planning framework generating behaviour that should take a system from a (probabilistic) initial state to an intended goal state. The states and domains are represented in DTPDDL, an extension to the well-establish planning language PDDL [10], which is described along with other details about the employed planning framework in Sec. 5 and is one of the key contributions to this paper. Making the deliberative layer domain-independent allowed us to reuse the very same framework and also the architecture schema in other systems that are confronted with a rather different task, such as learning new objects interactively from a human tutor [11].

Also part of the deliberation layer is the goal management framework [12, 11]. As we are aiming for a self-extending system the robot shall be motivated to extend its knowledge autonomously. Hence, a motivational framework continuously analyses the state of the world watching out for *knowledge gaps*, that can give rise to epistemic goals. Without going into detail, this can be thought of as general drives of the robot which task the planning system with new goals that yield some knowledge gain. For instance, being uncertain about the category of a room gives rise to a goal to be certain about it and similarly for other types of goals to be discussed later.

2.3 Conceptual Layer

While competences implement specific behaviours to gather knowledge, ignorant to the actual task or context this knowledge might be used for, the deliberative layer needs to consult this gathered knowledge in the context of a domain to employ the domain-independent planning algorithms. Hence, we need a way to put the competences into context and expose both domain knowledge and gathered knowledge to the deliberative layer. Here the conceptual layer comes into play. As stated above, we particularly aim to model and deal with the uncertainty that occurs in the real world. Hence our conceptual layer is inherently probabilistic – both in the represention of the beliefs as well as any background knowledge that is part of the domain model. Consequently, the conceptual layer (i) selects and translates information entities gathered through different competences, (ii) maintains relations between information entities and to the domain model, and (iii) unifies the gathered and background knowledge of the system into a probabilistic representation facilitating reasoning and planning. Said in simple terms, it mediates between the (strictly decoupled) deliberative layer and the competences, by relating the acquired information to the domain model and supporting reasoning about the resulting probabilistic relations through a chain graph formalism. This reasoning and further details of the conceptual layer will be given in Sec. 4.

Due to the strict decoupling of deliberation and the competences, the conceptual layer also has to mediate any planned action devised by the deliberative layer. An "Executor" component translates the domain-independent PDDL representation of planned actions into domain-targeted competence-level actions that are then executed and monitored by the Executor. There are further feedback channels and communication pathways in the architecture that have been omitted in Fig. 1(a) for clarity.

2.4 Platform

For clarity of the following sections, we briefly present the robot platform used to assess our approaches with. We implemented the system on our robot Dora, which is a based on a PIONEER 3DX platform with a custom-made superstructure accommodating a pan-tilt unit on which a MICROSOFT® KINECT™ and a POINT GREY CHAMELEON™ CCD USB camera are mounted. A HOKUYO laser scanner enables the robot to measure distances to objects in a range of 270° for a distance of up to four meters. Fig. 1(b) illustrates the construction of the robot. The pan-tilt unit is mounted so the robot can look on top of tables and similar surfaces to search for objects. The software system is running on a 2.6 GHz, 4Gb RAM Core2 Duo laptop mounted on the superstructure.

3 Competences

In the following sections we describe the different competences implemented in our architecture. Rather than explaining each of these in full technical detail we focus on the functional role these competences play in the context of our system. Hence, the emphasis lies on the structure and nature of the information they gather and expose, and the actions they provide. Further details can be found in the publications referenced in the respective sections. In this section we introduce the names of the actions (typeset in `typewriter` font) that the respective competences provided to the planner to ease understanding of Sec. 5 and 6.

3.1 Navigation & Map Building

The spatial representation is key to a mobile robot. It provides the basic symbols that the rest of the system, such as a planner, have to use to reason about and interact with the real world – and also mediates that interaction through actions (such as to move the robot's base to a position).

The task of the "Mapping & Navigation" competence (provided by the spatial subarchitecture) is to take in sensory data and distil this into functionally distinct, discrete, abstract units – as well as to receive commands on an abstract level and translate them into continuous-space control processes. The abstraction and discretisation process is important as it makes the representation tractable and more robust to changes in the world. Discretisation also reduces the number of states considered e.g. during the planning process.

The low-level sensory inputs to the subarchitecture are: 2D laser scans, 3D point clouds (from the Kinect™ depth sensor), odometry data from wheel encoders, and camera images. Outputs consist of wheel velocities and pan/tilt-commands to the camera assembly. Internally, the inputs are assembled into a consistent metric map used for navigation, obstacle avoidance and visual search (see Sec. 3.4). The abstraction provided to the rest of the system, however, is in terms of discrete *Places*.

The aim of this "Place"-based representation is to represent the world at the level of accuracy sufficient for performing required actions and robust localisation despite uncertainty and dynamic variations[13]. Here, each Place is based on a node in a navigation graph, as described previously in [14].

Places are also associated with *doorways*, i.e. narrow openings that separate different rooms in a map. A Place is annotated as a doorway using features of the 2D laser scan, and this property is used to group Places into rooms using non-monotonic reasoning (see [15]). Hence, the spatial representation provided by this layer features Places that are assigned to rooms. Places that are annotated as doorways do not belong to any room.

The system also represents paths between Places. The semantic significance of a path between two Places is the possibility of moving directly between one and the other. Space that has not yet been explored by the robot has no Places in it. Therefore, hypothesised Places are generated in unexplored locations near to extant Places. These hypothetical Places allow for reasoning about unknown space, and for planning and executing exploratory activities. They are annotated as *Placeholders* to distinguish them from actual Places, but are otherwise identically represented and interconnected. For an illustrative example see Figure 2.

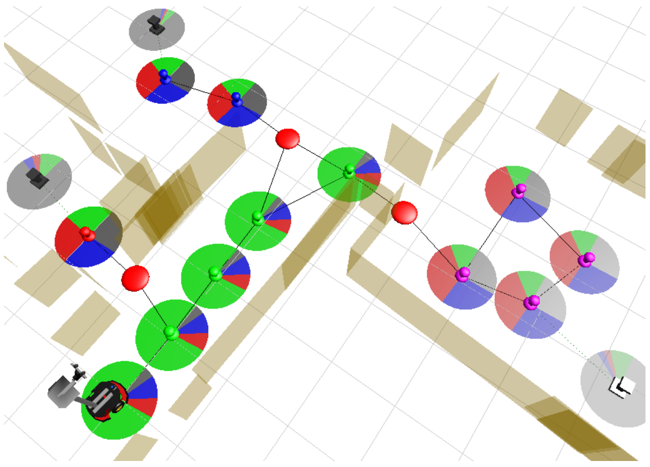


Figure 2: A Place map with several Places and three detected doors shown as red dots. Colours on circular discs indicate the probability of room categories as in a pie chart: i.e. the bigger the colour the higher the probability. Here green is *corridor*, red is *kitchen* and blue is *office*. Gray circles denote Placeholders, which represent unexplored space.

Besides providing the abstracted Places and Placeholders, and their connectivity information, to the higher reasoning modules in the system, the spatial subarchitecture also exposes concrete physical actions that the planner can perform on these abstractions. These actions are covered by a single planner primitive: `move`, which moves the robot from the Place where it is, to another neighbouring Place to which there exists a path. The same action is used for exploratory movement as well, by using a Placeholder as the goal Place. On the lower level, this single action encapsulates rotational and translational movement, local path finding and obstacle avoidance, as well as visual exploration (essentially, having a "look around" with cameras and Kinect$^{\text{TM}}$) upon reaching the goal Place in order to populate the metric maps. This action is used to explore unknown space as well as to move between rooms and to take up position for visual search. The simple `move` is complemented by a `move_direct` action which enables the robot to move to another Place that is connected to the current Place through another Place. This allows the robot to cover larger distances more quickly, as it doesn't necessarily have to travel exactly through a Place.

3.2 Place Categorisation

We want our robot to be able to learn and exploit the category of rooms (e.g. whether a room is a kitchen or an office). Based on the segmentation of the known space into rooms as outlined before, this competence aims to augment places with information that yields evidence about room categories. Therefore, it exploits the robot's various sensors to get such evidence about the category of the room a place belongs to. Here, we particularly aim not only to re-recognise rooms that the system has been trained for, but instead realise a competence that can provide evidence from general models that are not necessarily as accurate as specific ones, but still yield some useful evidence in unknown environments.

The different modalities to be analysed yielding the required envidence are the shape, size and appearance properties of places. The competence is realized as a purely passive (so it runs continuously in the background) process that analyses the laser data and images captured from the robot's CCD camera. Following [16], a small, discrete set of views is acquired at each place and two types of low level features are extracted: (a) geometric features extracted from laser range data for the purpose of shape and size classification for the room, and (b) global appearance features based on Composed Receptive Field Histograms obtained from second order normalised Gaussian derivative filters applied to the illumination channel at two scales. Based on those features, independent categorical models are built for the shape (e.g. elongated or rectangular), size (e.g. small or large) and appearance (e.g corridor-like or office-like) properties.

To provide sufficient robustness and tractability in the presence of noisy, high-dimensional information, we use non-linear kernel-based discriminative classifier models, namely Support Vector Machines, as proposed in [17]. The models are trained from sequences of images and laser range data recorded in multiple instances of rooms belonging to different categories and under various illumination settings (during the day and at night). By including several different room instances in the training set, the acquired model can generalise sufficiently well to provide categorisation rather than instance recognition. In order to measure the uncertainty associated with the generated hypotheses, confidence measures are derived from the distances between the classified samples and discriminative model hyperplanes [17]. The accumulated confidences gained from the SVM models across views are normalised to gain probabilities.

3.3 Object-Room Co-occurrence

As outlined in Sec. 1, competences not only gather knowledge from the current situation or environment, but they can also extend to query information available online. An example implemented in our robot Dora is the competence to gather commonsense conceptual knowledge that encapsulates how likely it is that particular objects can be found in specific locations, e.g. that sinks and faucets are more common in kitchens and bathrooms than in living rooms, and that computers occur more frequently in offices than in corridors. Our approach is to leverage commonsense knowledge available through the world wide web to yield object-location co-occurrence priors (normalised values between 0.0 and 1.0), as described in [2]. Here, we only provide a brief summary of the approach.

The Open Mind Indoor Commonsense[1] locations database (henceforth referred to as OMICS-L) provides more than 5,800 user-given associations between common everyday objects (ca. 2,900 unique types) and their typical locations (ca. 500 unique types). This provides us with a rich set of objects and locations that are relevant for intelligent mobile indoor robots. In total, we can generate a matrix of approx. 1.5 million unique object-location pairs. In order to quantify the prior likelihoods of each of these pairs, we queried an image search engine[2] for the number of hits that are returned when searching for these object-location pairs and compare it to the number of hits for the locations alone. The normalization of the acquired raw frequencies is explained in more detail in [2].

In the present system, the co-occurrence priors are one kind of probabilistic knowledge that is exposed to the system. Since the amount of data is much too large to be used as a whole, our approach is to make specific probabilistic facts available to the system on demand. The facts are queried in such a way that probabilistic co-occurrence a priori knowledge is available for those objects for which visual models have been trained in advance (see Sec. 3.4), and for those kinds of rooms that can be categorised by the system (see Sec. 3.2). Currently, these co-occurrence priors are currently acquired in an off-line process and not actively acquired by the deliberative layer. However, the extension to online processing is straightforward and doesn't yield any major scientific insights.

The resulting co-occurrence priors constitute quantitative *commonsense* knowledge that endow the system with expectations about what can be found where in order to plan and act efficiently in partially unknown environments and with uncertain knowledge about it. It might, however, not necessarily reflect the distribution of objects in the current environment – and most likely won't. It is therefore used as background knowledge when judging where an action might most likely have its intended effect (such as finding the cornflakes when searching for cornflakes in a particular location), and not as a knowledge base of the current situation. Details about the discrimination between background and instance knowledge will be presented in Sec. 4.

3.4 Viewpoint-based Object Search

As stated above, the "Object-Room Co-Occurrence" competence yields background knowledge about the location of certain types of objects in certain categories of rooms. However, this knowledge is generic and does not necessarily apply to the current situation. In order to actually localise an object or verify its existence the robot must be able to conduct a *visual search* for the object in question. Active Visual Search is a research challenge in its own right and is addressed by a number of works [18, 19, 20]. In this paper, we employ some state-of-the-art algorithms to endow our robot with a competence to search for objects. This competence enables the robot to, obviously, achieve goals to find certain objects in the world, and gain evidence about room categories. Consulting the background knowledge obtained through the "Object-Room Co-Occurrence" competence, the existence of a microwave in a room could be seen as relevant evidence to the question of whether the room is a kitchen, for instance.

The approach to object search in our system is based on the inspection of a set of promising *viewpoints*. A viewpoint is defined as a 4-tuple $V_i = (\Pi_i, \alpha_i, \beta_i, P_i(O))$, comprising the place Π (in fact, a 2D position in the map), the view direction given

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

[2] `http://images.bing.com`, September-October 2010

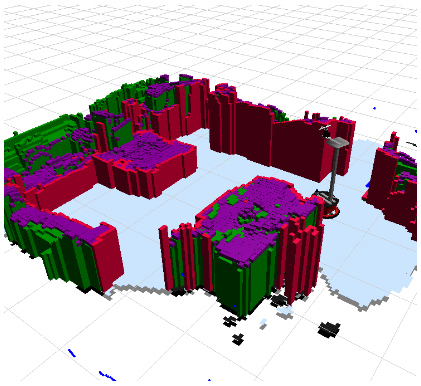
as two angles for panning α and tilting β, and the probability of seeing the object O in question in that particular viewpoint. Hence, the viewpoints form *cones* in the 3D representation. The driving idea behind the approach is that the robot should not randomly look everywhere, but shall exploit the knowledge it has about the environment through other competences (e.g. the metric map) to optimise the search, i.e. inspect a set of promising viewpoints that maximise the probability of seeing the object first. In order to allow our system to reason about the benefit of inspecting different viewpoints, object search is not realised as one monolithic action, but divided into two major parts: "viewpoint generation" (the planner action `create_cones_in_room`) and "viewpoint processing" (`process_conegroup`). This allows the deliberation layer to sequence the inspection of the viewpoints in the context of other knowledge gathering actions it could schedule.

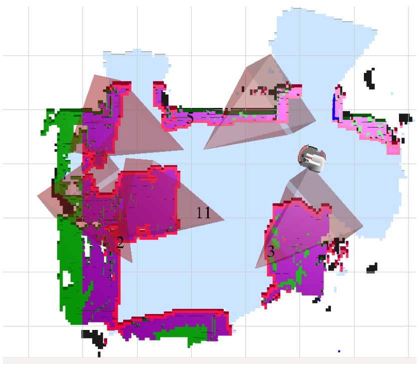
In order to generate viewpoints to search a given room, we employ a similar strategy to [21]. The search environment is tessellated into cells with each cell containing the occupancy of the cell (i.e. UNKNOWN, OCCUPIED or FREE) and the probability of the target object's center being in this cell. This amounts to a 3D probability distribution over the search space for a target object. In this work, the 3D probability distribution is initialised by assigning a uniform probability to the region on top of known 3D obstacles, see Figure 3(a).

The positions for potential viewpoints are picked from the center of places and different directions are sampled from each place. Each potential viewpoint is then evaluated on the basis of its probability value. The potential viewpoint with the highest probability is retained and the process is repeated with its probability value subtracted from the portion of the map that it covers until more than a predefined of amount of probability mass is covered (usually about 95%). For a more detailed discussion on the view planning we refer the reader to our previous work [22, 23].

Viewpoints that can be processed by simply turning the camera (using the pan-tilt unit, i.e. without the robot having to move) are grouped together. The reason for this grouping is to minimize movement while processing viewpoints. Each group corresponds to a `process_conegroup` action and the probability of finding the object after executing the action is the sum of the probability values of its viewpoints; see Figure 3(b). Upon receiving a command to execute a `process_conegroup` action, the robot moves to the associated place and orients itself towards the most central viewpoint. It then turns its camera in the direction of each viewpoint in turn, whereupon the object detection algorithm is triggered. The robot captures a monocular image from its camera and runs SIFT-based recognition [24] for the object being searched for. Upon successful detection the estimated 3D pose is computed and the object's model is put into the grid map as an obstacle, so that it will occlude future views properly. If the object which is currently being searched for is not detected in a view, a probability update is performed on the space encompassed by that viewpoint, less any parts that are occluded. This reduces the posterior probability density within the viewpoint, and increases it elsewhere in the map, while the total probability that the object is in the room decreases.

Note that this detection approach requires pre-trained models of any object we want to search for. The domain model features the information about which objects can be searched for so that the planner can take this limitation into account when devising plans.


(a) Prior probability distribution for an object in a room (purple), based on obstacles (green).



(b) Examples of viewcones with probabilities.

Figure 3: Creating viewpoint actions for object search

3.5 Dialogue

We have discussed competences that exploit the web as a knowledge source (cf. Sec. 3.3) and to inspect the environment with the robot's own sensors (e.g. searching for an object or gaining evidence about the category of rooms). Now, we shall turn towards the exploitation of humans as knowledge sources. The general principle does not at all differ from other competences to gather information. We embrace asking a human for information as an uncertain process like we do for others. We model the probability of finding a human in a room who could provide us with the information we are looking for. This probability obviously could be dependent on the category of the room and the question we are about to ask. Hence, the questions of if, when, and what to ask to gain some new knowledge have to be decided by the deliberative layer.

Put simply, this competence enables our robots to initiate and conduct situated dialogues for interactive learning. The potential for people to know and provide certain kinds of information about the environment is encoded in the planning domain. Asking questions is triggered by the planner by making it an explicit knowledge-gathering action. While its postconditions assert that after interpreting the human's answer the robot possesses the knowledge in question, its preconditions ensure that the robot follows certain social rules when engaging in such a conversation. We make use of the dialogue framework presented in [11]. As the framework is largely domain-independent, only a few modifications to the domain models were required for its deployment in the described scenarios.

Dialogue design In the context of the present scenario, we focus on robot-initiated dialogues. Based on the current state of the knowledge base, the planning system (cf. Sec. 5) can decide that asking a human will provide it with information that helps it better achieve its goals. Before the robot can talk to a person, however, it first needs to find somebody to talk to. Once a person is found – the details of this action are described in Sec. 3.6 – the robot first needs to properly engage in a dialogue with that person (planner action `engage`) before asking the human any specific questions. This engagement is established through an opening turn by greeting the human. After the human acknowledges the opening of a dialogue by, in turn, greeting the robot, the planner is permitted to request the dialogue system to ask the appropriate knowledge gathering or clarification questions.

Dialogue generation Asking a *polar question*, such as "is this room a kitchen?" (an example of the `ask-for-category-polar` action), is a possibility for hypothesis verification. *Open questions* like "what kind of room is this room?" allow the robot to fill knowledge gaps for which not enough prior knowledge or an insufficient level of certainty about a hypothesis exist. Referring expressions are generated using the approach described in [25]. Our example contains the referential description "this room." By the very nature of that situation, the robot does not possess any more specific, certain knowledge about the room's category (other than that it is some kind of room), nor does it know what the human knows about the room. As a consequence the robot asks "is this room a kitchen?" rather than unnatural-sounding and unplausible questions like "is the kitchen a kitchen?". The language production subsystem makes use of the MARY text-to-speech system [26].

Dialogue interpretation In task-oriented dialogues between a human and a robot, there is more to dialogue than just understanding words. The robot needs to understand what is being talked about, but it also needs to understand why it was told something. In other words, what the human *intends* the robot to do with the information in the larger context of their joint activity.

Therefore, understanding language can be phrased as an *intention recognition* problem: given an utterance from the human, how do we find the intention behind it? We extend Thomason and Stone's abductive account of language understanding, planning and production [27], in which agents actively monitor and maintain common ground, and to this end they attempt to *abductively* recognize the others' *intentions* as explanations of their observed (linguistic) behaviour.

Our extension of this approach lies in explicitly modelling the knowledge gaps that inevitably arise in such an effort due to uncertainty and partial observability. The approach is based on generating partial hypotheses for the explanation of the observed behaviour of the human agents, under the assumption that the observed behaviour is intentional. These partial hypotheses are defeasible and conditioned on the validity (and eventual verification) of their assumptions. The abductive proof procedure is an extension of Brenner and Nebel's work in continual automated planning [28]. Their notion of *assertion* allows the system to reason about information not present in the knowledge base, thereby addressing the need for reasoning under the open-world assumption. The details of our approach are described in [29].

In our system, this abductive intention recognition process is used to interpret the human's answers to the question that the robot asks. For instance, answering "yes" to a polar question expresses the user's intention that the robot be more certain about the fact in question, which then typically results in this particular statement (e.g. the current room's category is 'kitchen') being added as a (strong) evidence to the robot's relational model (cf. Sec. 4).

Example dialogue The following is a typical example of a robot-initiated dialogue for knowledge gathering and verification. Below we focus on the aspects of dialogue interpretation. The systemic effects underlying such a dialogue are explained in more detail in Section 6.2.

- robot: "hello human" *opening engagement*

- human: “hello dora” *dialogue interpretation yields that the dialogue has been successfully opened*
- robot: “ok” *backchannel response about successful interpretation of the user’s utterance*
- robot: “is this room a kitchen?” *verification question triggered by the planner*
- human: “yes” *dialogue interpretation infers the human’s intention that the proposition underlying the preceding question be added as factual knowledge*
- robot: “ok” *backchannel response about successful interpretation of the user’s utterance*

The robot’s utterances “hello human” and “is this room a kitchen?” are triggered by the planner. The response feedback “ok” originates in the dialogue interpretation. It signals (cf. [30]) that the robot was able to come up with an interpretation of the human’s utterance that is compatible and consistent with its belief state. While in this system we have chosen a simple verbal feedback (“ok”), this response could be replaced with a more natural-sounding vocal backchannel signal (“hmm”, “uh-huh”) as described by Schröder *et al.* [31]. Their listener vocalization module is based on the same text-to-speech system as the one used in our system, cf. [32].

3.6 Person Search

As defined in the previous section about the dialogue competence, finding a person to talk to is an enabling competence for actually engaging in a dialogue. In order to allow our system to find persons, we implemented a active person search competence based on input (image and depth information) from the MICROSOFT® KINECT™. It is modelled as a stochastic action `look-for-person` that has a probabilistic effect to see a person at a place if s/he is within communication range to that given place (currently set to $3m$). Whenever that action is triggered, the robot will turn in that place to acquire a random orientation. Then it commands the pan-tilt unit to scan the area around the robot stepping in angles of 80% of the aperture angle of the Kinect™ to cover the whole scan area with some overlap between views. For each view acquired, the OpenCV face detector (using cascades of simple detectors for simple Haar-like features [33]) is run to yield a number of potential faces. To reduce the number of false positives, those hypotheses are filtered using a simple heuristic: It requires the distance to that face, estimated on the assumption of a constant size for human faces, to be within a 50cm margin around the median depth reported by the Kinect™ for the area of the face. Upon successful detection, the detected person is associated with the nearest place and the probability of that respective person being at that place is revised accordingly, yielding the update of the belief state in the conceptual layer. In our current model we expect there to be one or no person within a room with a certain default probability being encoded in the domain model of the conceptual layer (cf. concept “Person” and its relations in Fig 4).

4 Conceptual Layer

The conceptual layer is the representation and reasoning layer unifying the knowledge in the system. It comprises *gathered* knowledge, actively acquired through the competences presented above to realise self-extension, and predefined *conceptual knowledge* as part of the overall domain model which is also part of the conceptual layer. For a mobile robot such as ours, it also realises the highest layer of the qualitative spatial framework presented in [34, 16], where it has been referred to as the *conceptual map*

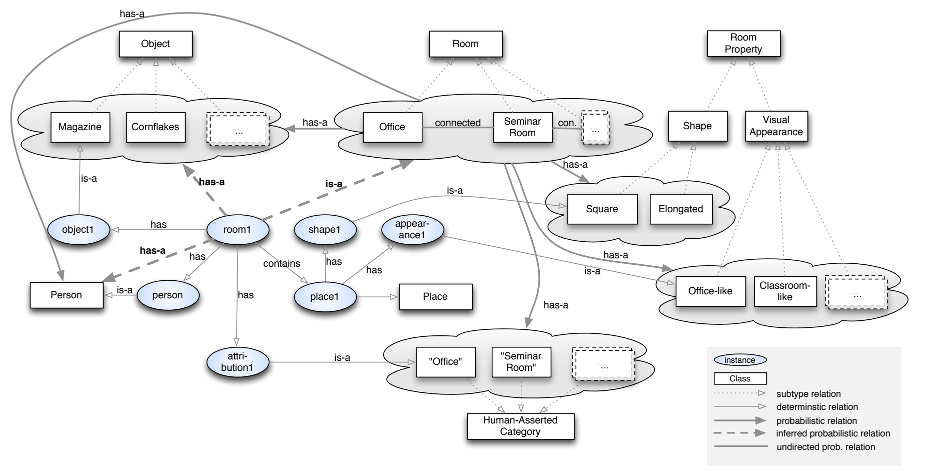


Figure 4: A visualised excerpt of the ontology of the conceptual layer. It comprises knowledge about concepts (rectangles) and the relations between those concepts and instances of spatial entities (ellipses) of which only an example are shown here. Where relations are probabilistic (see legend) they refer to a set of concepts which represent possible assignments to the respective probabilistic relation. The model features both directed (shown with arrow heads) and undirected relations (the "connected" relation between different subtypes of Room is an example of the latter). This representation is compiled into a chain graph representation (see Fig. 5) to draw inference about unseen relations.

due to its strong emphasis of spatial information. It accommodates a relational representation, storing conceptual knowledge as relations between concepts and capturing *instance knowledge* as relations between either instances and concepts, or instances and other instances. In order to account for the uncertainty of both our domain knowledge as well as the gathered instance knowledge, relations are inherently uncertain and thus represented using a probabilistic framework. Hence, the conceptual layer also features probabilistic inference of not directly observed instances and relations. Taken together, the conceptual layer forms the robot's beliefs about the world and unifies these beliefs into a probabilistic belief state accessible to the deliberative layer.

4.1 Relational Model

An excerpt of the representation in the conceptual layer is visualised in Fig. 4. Relations are either *predefined* as part of the domain model, *acquired* through one of the competences, or *inferred* through inference in the conceptual layer. Uncertain knowledge is represented as probabilistic relations. However, not all relations are necessarily probabilistic; the representation also supports deterministic relations. In general, a non-existent relation can therefore be thought of as having probability 0. An acquired relation is one that is grounded in observations and generated as a result of a perceptual process. Predefined relations are given (and quantified if they are probabilistic) as part of a fixed ontology of default knowledge. Overall, the representation defines a taxonomy of concepts and associations between instances and concepts using hyponym relationships ("is-a"). Then, directed relations ("has-a") are used to describe, e.g., properties of room categories in terms of spatial properties, such as shape, size or appearance, and objects. Finally, there is also support for undirected associative relations to, e.g., represent connectivity between rooms; to encode knowledge such as that a corridor is likely to be next to a bathroom, but a bathroom is unlikely to neighbouring a kitchen.

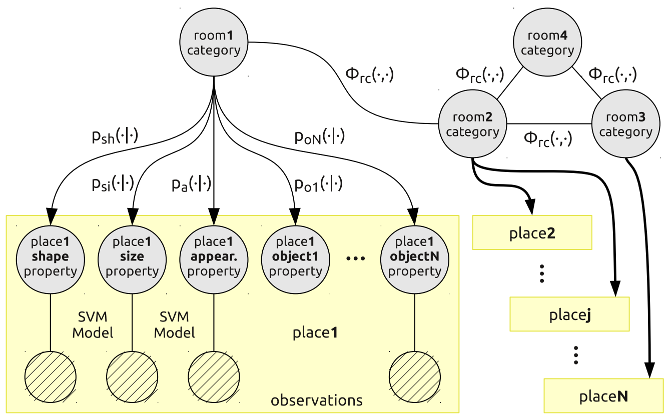


Figure 5: Structure of the chain graph model compiled from the relational model. The vertices represent random variables. The edges represent the directed and undirected probabilistic relationships between the random variables. The textured vertices indicate observations that correspond to sensed evidence.

Probabilistic relations allow the expression of statistical dependencies and uncertainty. An example is the “has-a” relation linking subtypes of “Room” to subtypes of “Object”. It constitutes a probabilistic relation that is quantified by the competence “Object-Room Co-occurrence” as described in Sec. 3.3 and can represent the probability that generally, e.g., sinks and faucets are more common in kitchens and bathrooms than in living rooms. It shall be noted that this is background knowledge that does not necessarily apply to the current situation. However, the “has-a” relation for a specific room instance (e.g. for “room1” in Fig. 4 having a certain type of object in it) can then be inferred from the knowledge in the conceptual layer taking into account other evidence such as the knowledge about specific “Room Properties”.

4.2 Drawing Inferences

In order to allow (Bayesian) inference in the conceptual layer, the relational representation is compiled into a *chain graph* representation [16], whose structure is adapted on the fly reflecting the state of the underlying belief models. Chain graphs provide a natural generalisation of directed (Bayesian Networks) and undirected (Markov Random Fields) graphical models, allowing us to model both “directed” causal (such as “is-a” relations) as well as “undirected” symmetric or associative relations (such as connectivity of rooms discussed above). The use of a chain graph allows us to model circular dependencies originating from possible loops in the topological graph, as well as direct use of the probabilistic relations between the concepts. In our implementation, chain graph inference is event-driven, triggered by updates to the belief model (cf. Fig. 1(a)). For example, if an appearance property, or object detection alters the probability of a relation, inference proceeds to propagate the consequences throughout the graph. In our work, the underlying inference is approximate, and uses the fast Loopy Belief Propagation [35] procedure.

An exemplary chain graph corresponding to the relational model shown in Fig. 4 is presented in Fig. 5. Each discrete place instance is represented by a set of random variables, one for each class of relation linked to that place. These are each connected to a random variable over the categories of rooms, representing the “is-a” relation between rooms and their categories in Fig. 4. Moreover, the room category variables are connected by undirected links to one another according to the topological map. Here, the potential functions $\phi_{rc}(\cdot,\cdot)$ describe the type knowledge about the connectivity of rooms of certain categories (e.g. that kitchens are more likely to be connected to corridors than to other kitchens).

The remaining variables represent: shape and appearance properties of space as observed from each place; and the presence of objects. These are connected to observations of features extracted directly from the sensory input. As explained in Sec. 3.2, these links are quantified by the categorical models of sensory information. Finally, the distributions $p_s(\cdot|\cdot)$, $p_a(\cdot|\cdot)$, $p_{o_i}(\cdot|\cdot)$ represent the common sense knowledge about shape, appearance, and object co-occurrence, respectively. They allow for inference about other properties and room categories, e.g. that the room is likely to be a kitchen, because you are likely to have observed cornflakes in it.

With all these features and abilities the conceptual layer provides a unified view to obtain a probabilistic state space suitable for the deliberative layer. It comprises the predefined domain knowledge and the gathered instance knowledge compiled into a consistent Bayesian representation that is then translated into a DTPDDL representation. How exactly the uncertain knowledge and state is represented, and then planned with, is subject to next section.

5 Switching Planner

The planning subarchitecture is responsible for coordinating the overall system behaviour. The use of a domain independent planner provides us a relatively easy way to extend the capabilities of the system and allow for a wider range of tasks. As the tasks we are interested in involve knowledge gathering in partially unknown environments, we employ a decision theoretic (DT) planner that can exploit the probabilities provided by the conceptual layer. For example, when searching for an object the planner may decide to first ask a human for the type of the current room (i.e. scheduling an `ask-for-category-polar` action as offered by the dialogue competence introduced in Sec. 3.5), if this information is strongly correlated with the probability of finding the object there. As full decision theoretic planning is not feasible in large environments and as it is difficult to model *open worlds* as a POMDP, we use a classical (sequential) planner in a *continual planning* framework to decide on the high level strategy and use the DT planner in instances where sensing is required. The classical planner can perform limited reasoning about probabilities, so it may decide to `move` to a room that has a high likelihood of containing the target object and perform a search there. Only once the robot has reached that room will the DT planner take over to decide in detail which sensing actions should be performed.

5.1 Representing Uncertain Knowledge

There are three representations for uncertain knowledge used by the planning system:

- *assumptive actions* are used to model probabilistic relations between facts.
- The *factored planning state* represents a *belief state* as a tree of facts similar to PPDDL[36].
- The *flat belief state* is a probability distribution over all possible states.

As the third representation is only used internally by the DT planner, we will only describe the first two in more detail. The structure of the planning problems as described here is somewhat simplified from the full version, which can be found in appendix A.

The syntax for describing an initial state distribution is taken verbatim from PPDDL. That distribution is expressed in a tree-like structure of terms. Each term is either: (1) atomic, e.g., a state proposition such as $(=$ (related-to milk) room0$)$; (2) probabilistic, e.g., (probabilistic $\rho_1(T_1)..\rho_n(T_n)$) where T_i are conjunctive; or (3) a conjunct over probabilistic and atomic terms. The root term is always conjunctive, and the leaves are atomic. For example, a simplified object search could have:[3]

```
(:init
    (= (is-in Robot) room0)
    (probabilistic
         0.4 (and (= (category room0) kitchen)
                  (probabilistic .7 (= (related-to cup) room0))
                  (probabilistic .8 (= (related-to milk) room0)))
         0.4 (and (= (category room0) office)
                  (probabilistic .5 (= (related-to cup) room0))
                  (probabilistic .1 (= (related-to milk) room0)))
         0.2 (and (= (category room0) corrdior)
                  (probabilistic .2 (= (related-to cup) room0)))))
```

The interpretation is given by a *visitation* of terms: An atom is *visited* iff its conjunctive parent is visited, and a conjunctive term is visited iff all its immediate subterms are visited. A probabilistic term is visited iff its conjunctive parent is visited, and exactly one of its subterms, T_i, is visited. Each visitation of the root term according to this recursive definition defines a starting state, along with the probability that it occurs. The former corresponds to the union of all visited atoms, and the latter corresponds to the product of ρ_i entries on the visited subterms of probabilistic elements. Making this concrete, the above example yields the following flat distribution (using $\perp$ as *undefined)*:

Prob.	(is-in Robot)	(category room0)	(related-to milk)	(related-to cup)
.224	room0	kitchen	room0	room0
.096	room0	kitchen	room0	$\perp$
.056	room0	kitchen	$\perp$	room0
.024	room0	kitchen	$\perp$	$\perp$
.02	room0	office	room0	room0
.18	room0	office	room0	$\perp$
.02	room0	office	$\perp$	room0
.18	room0	office	$\perp$	$\perp$
.04	room0	corridor	$\perp$	room0
.16	room0	corridor	$\perp$	$\perp$

Assumptive actions are used to allow a more compact representation of probabilistic background knowledge. As an alternative to describing the complete initial state using grounded probabilistic statements, assumptions allow us to represent relationships between facts in a *lifted* form.

The following assumption is an description on how room categories relate to the existence of objects using conceptual knowledge:

[3]In PDDL, (:init $T_1..T_n$) expresses the conjunctive root of the tree – i.e., the root node (and $T_1..T_n$). Also, we shall write p, rather than (and p), for conjunctive terms that contain a single atomic subterm.

```
(:assume object-in-room
   :parameters (?l - label ?r - room ?c - category ?o - visualobject)
   :precondition (and (= (category ?r) ?c)
                      (= (label ?o) ?l))
   :effect (probabilistic (prob-inroom ?l ?c)
                          (assign (related-to ?o) ?r)))
```

It should be read as: if the room `?r` is of category `?c`, and a value `(prob-inroom ?l ?c)` is provided by the conceptual knowledge store, then an object `?o` of type `?l` is in `?r`.

Both representations are equivalent: a planning task that contains lifted assumptions can be converted to a DTPDDL state distribution, and a tree of `probabilistic` statements can be turned into a set of grounded assumptions. As an input to our system we use a combination of both representations. Some systems provide us with direct probability distributions over facts (e.g. the category of a room). Those are represented as a `probabilistic` expression in the initial state description. These distribution are the base on which further assumptions can be made, which are usually given in a lifted format.

5.2 Modelling Observations

For the decision theoretic planner, we require a representation of sensing that can be used to model POMDPs. We developed DTPDDL, an extension of PPDDL, that can express probabilistic models of the sensing consequences of acting, to quantitatively capture unreliability in perception. There are straightforward compilations from problems expressed in DTPDDL to flat state-based (and propositionally factored) representations of the underlying decision process. Although similar to the POND input language [37], DTPDDL distinguishes itself by explicitly treating state and perceptual symbols separately, and by providing distinct declarations for operators (i.e., state model) and senses (i.e., observation model). In this last respect, DTPDDL admits more compact domain descriptions where sensing effects are common across multiple operators.

To model sensing capabilities, we have operator-like "sense" declarations, with preconditions expressed using state and action symbols, and uniformly positive effects over perceptual symbols. For example, where *look-for-object* is the operator that applies an object detection algorithm at a specific place, an *object search* task will have:

```
(:sense vision
   :parameters (?r - robot ?v - visual-object ?l - location)
   :execution  (look-for-object ?r ?v ?l)
   :precondition (and (= (is-in ?r) ?l) )
   :effect (and (when (= (is-in ?v) ?l)
                      (probabilistic .8 (= (o-is-in ?v) ?l)))
                (when (not (= (is-in ?v) ?l))
                      (probabilistic .1 (= (o-is-in ?v) ?l)))))
```

i.e., there is a 10% *false positive* rate, and 20% probability of a *false negative*. This representation allows us to represent actions that have multiple independent observational effects.

5.3 Switching Continual Planning

Our *switching* planner is based on the continual planning paradigm: A continual planner creates a plan containing assumptions that are compatible with the current observed state and starts executing the plan. If an action fails to execute or new observations indicate that the current plan will no longer reach the goal, the planner *replans* from the new state and repeats the process until either the goal is reached or no further plan can be found. A formal outline of the system is given in Algorithm 1. This system *switches* because planning and plan execution proceed in interleaved sessions in which the *base planner* is either *sequential* or *decision-theoretic*. The initial session is sequential (Alg. 1, Line 7), which begins when a DTPDDL goal is given to the planner and a DTPDDL problem description is given to the system. Given in Algorithm 2, a sequential session computes a serial plan that corresponds to one execution-*trace* in the underlying decision-process (Alg. 2, Line 4). That trace is a reward-giving sequence of process actions and *assumptive* actions. Each *assumptive* action corresponds to an assertion about some facts that are unknown at plan time – e.g. that a box of cornflakes is located on the corner bench in the kitchen. The trace specifies a plan and characterises a *deterministic approximation* (see [38]) of the underlying process in which that plan is valuable. Traces are computed by a cost-optimising classical planner which trades off action costs, goal rewards, and determinacy. Execution of a trace proceeds according to the process actions in the order that they appear in the trace (Alg. 2, Lines 8-18). If, according to the underlying belief-state, the outcome of the next action scheduled for execution is not predetermined above a threshold (Alg. 2, Lines 13), then the system switches to a DT session (Alg. 1, Line 11).

Algorithm 1 Switching Continual Planner

```
1: Input:
     • DTPDDL problem description $\Pi$.
2: Output:
     • success or failure
3: loop
4:   if CHECKGOAL($\Pi$) then
5:     return success
6:   end if
7:   $\Pi, \pi \leftarrow$ SEQUENTIALSESSION($\Pi$)
8:   if $\pi$ = failure then
9:     return failure
10:  else if $\pi \neq []$ then
11:    $\Pi \leftarrow$ DTSESSION($\Pi, \pi$)
12:  end if
13: end loop
```

The pseudocode for a DT session is given in Algorithm 3. Because online DT planning is impractical for the size of problem we are interested in, DT sessions plan in a small abstract problem defined in terms of the trace from the proceeding sequential session. The pseudocode for the procedure that yields that abstraction is given in Algorithm 4. The abstract state-space is characterised by a limited number of propositions, chosen because they relate evidence about assumptions in the trace. To allow the DT planner to judge assumptions from the trace, we add *disconfirm* and *confirm* actions to the problem for each of them (Alg. 3, Line 2). These model a relatively small reward/penalty if the corresponding judgement is true/false. If a judgement action is scheduled for execution, then the DT session exits (Alg. 3, Line 5), and a new sequential session begins according to Algorithm 1.

5.3.1 Sequential Sessions

In this paper we restrict our attention to deterministic-action POMDPs in which all state uncertainty is expressed in the `(:init)` declaration. This declaration is used by our approach to define the starting state for sequential sessions, and the set of assumptive actions available to sequential planning. The resulting deterministic problem model, Π^D (Alg. 2, Line 3), is the problem solved by a classical planning during the session.[4] For a sequential session the starting state corresponds to the set of facts that are true with probability 1. Continuing our example, that starting state is the singleton:

$$s_0 \equiv \{(= (\texttt{is-in Robot})\ \texttt{kitchen})\}.$$

To represent state assumptions in Π^D we augment the problem posed during a sequential session with an *assumptive action* $\mathcal{A}^\circ(\rho_i; T_i)$ for each element, $\rho_i(T_i)$, of each probabilistic term from `(:init)`. Here, $\mathcal{A}^\circ(\rho_i; T_i)$ can be executed if no $\mathcal{A}^\circ(\rho_j; T_j)$, $j \neq i$, has been executed from the same probabilistic term, and, either (`probabilistic` ..ρ_i (T_i)..) is in the root conjunct, or it occurs in T_k for some executed $\mathcal{A}^\circ(\rho_k; T_k)$. We also add constraints that forbid scheduling of assumptions about facts after actions with preconditions or effects that mention those facts. For example, the robot cannot assume it is plugged into a power source immediately after it unplugs itself. Executing $\mathcal{A}^\circ(\rho_i; T_i)$ in a state s effects a transition to a successor state s^{T_i}, the union of s with atomic terms from T_i, and of course annotated with auxiliary variables that track the applicability of assumptive actions. For example, consider the following sequential plan:

$\mathcal{A}^\circ(.4; (= (\texttt{category room0})\ \texttt{kitchen}));$
$\mathcal{A}^\circ(.8; (= (\texttt{related-to milk})\ \texttt{room0}));$
$\mathcal{A}^\circ(.7; (= (\texttt{related-to cup})\ \texttt{room0}));$
`(look milk room0)`; `(look cup room0)`;
`(report milk room0)`; `(report cup room0)`;

Applying the first action in s_0 yields a state in which the following facts are true:

$$\{(= (\texttt{is-in Robot})\ \texttt{room0}), (= (\texttt{category room0})\ \texttt{kitchen})\}$$

In the underlying belief-state, this is true with probability 0.4. The assumed state before the scheduled execution of action `(look milk room0)` is:

$$\{(= (\texttt{is-in Robot})\ \texttt{kitchen}), (= (\texttt{category room0})\ \texttt{kitchen}), \\ (= (\texttt{related-to milk})\ \texttt{room0}), (= (\texttt{related-to cup})\ \texttt{room0})\}$$

Which is actually true with probability 0.224 according to the underlying belief.

To give an optimisation criteria for sequential sessions we model each $\mathcal{A}^\circ(\rho_i; T_i)$ probabilistically, supposing that its application in state s effects a transition to s^{T_i} with probability ρ_i, and to $s^\perp$ with probability $1 - \rho_i$. State $s^\perp$ is an added sink. Taking ρ_i to be the probability that the i^{th} sequenced action, a_i, from a trace of state-action pairs $\langle s_0, a_0, s_1, a_1, .., s_N\rangle$ does not transition to $s^\perp$, then the optimal sequential plan has value:

$$V^* = \max_N \max_{s_0, a_0, .., s_N} \prod_{i=1..N-1} \rho_i \sum_{i=1..N-1} \mathsf{R}(s_i, a_i),$$

[4]Without a loss of generality we also suppose that actions do not have negative preconditions.

In our system, we have a very specific reward function that is geared towards use in a classical planner: Each action incurs a negative reward that is proportional to the action's costs: $\mathsf{R}(s_i, a_i) = -\mathrm{cost}(a_i)$. Additionally, a fixed positive reward is awarded for reaching the goal. The height of the goal reward determines whether the planner should prefer cheap but unlikely plans (low goal reward) or more expensive but more likely ones.

Algorithm 2 SEQUENTIALSESSION()

```
1: Input:
     • DTPDDL problem description $\Pi$, including initial belief-state $b_0$ expressed
       as an (:init) term
2: Output:
     • DPTDDL state resulting from the execution of actions.
     • Plan prefix $\pi$ that requires decision theoretic planning or *failure* if the prob-
       lem is unsolvable.
3: Initialise:
     • Determinisation $\Pi^D$ of $\Pi$ with assumptive actions $\mathcal{A}^\circ$
4: CLASSICALPLANNER produces a sequential plan $\pi = [a_1, a_2, ..., a_n]$ for $\Pi^D$
5: **if** $\pi = \emptyset$ **then**
6:     **return** $\Pi$, *failure*
7: **end if**
8: **for** $a_i \in \pi$ **do**
9:     **if** $\neg$CHECKAPPICABILITY$(\pi, \Pi)$ **then**
10:        **return** $\Pi$, []
11:    **else if** $a_i \in \mathcal{A}^\circ$ **then**
12:        – Ignore action $a_i$
13:    **else if** $\exists p \in \mathtt{pre}(a_i)$ s.t. $\mathsf{Pr}(p) < 0.95$ **then**
14:        **return** $\Pi, [a_1, \ldots, a_i]$
15:    **else**
16:        $\Pi, o \leftarrow$ EXECUTEACTION$(a_i)$
17:    **end if**
18: **end for**
19: **return** $\Pi$, []
```

5.3.2 DT Sessions

When, during a sequential session, an action is scheduled whose outcome is uncertain according to the underlying belief-state (Alg. 2, Line 13), planning switches to a DT session. This plans for *small* abstract processes defined according to the action that triggered the DT session, the assumptive actions in the proceeding trace, and the current belief-state. Targeted sensing is encouraged by augmenting the reward model to reflect a heuristic value of knowing the truth about assumptions. In detail, all rewards from the underlying problem are retained. Additionally, for each *relevant* assumptive action

$\mathcal{A}^\circ(\rho_i; T_i)$ (see Def. 1) in the current trace π, we have a *disconfirm action* $\mathcal{A}^\bullet(\rho_i; T_i)$ so that for all states s:

$$\mathsf{R}(s, \mathcal{A}^\bullet(\rho_i; T_i)) = \begin{cases} \$(T_i) & \texttt{if } T_i \not\subseteq s \\ \hat{\$}(T_i) & \texttt{otherwise} \end{cases}$$

where $\$(T_i)$ (resp. $\hat{\$}(T_i)$) is a small positive (negative) numeric quantity which captures the utility the agent receives for correctly (incorrectly) rejecting an assumption (Alg. 3, Line 2).

Definition 1 *(Relevant Assumption) Given a problem Π with belief b_0 and a sequential-plan prefix $\pi = [a_1, .., a_N, \tilde{a}]$ with switching action $\tilde{a}$, the set of* relevant assumptions *in the prefix corresponds to the set of assumptive actions, denoted $\mathcal{A}^{\circ R}$, given by:*

$$\begin{aligned} \mathcal{A}^{\circ R} \equiv \{a | a \in \pi \cap \mathcal{A}^\circ, \\ \emptyset \not\equiv (\ \mathit{add}(a) \cup \mathit{delete}(a)) \cap \mathit{pre}(\tilde{a}), \textit{ or} \\ \exists s.b_0(s) > 0, \exists \kappa \in K(\tilde{a}, s) \textit{ s.t.} \\ \emptyset \not\equiv (\ \mathit{add}(a) \cup \mathit{delete}(a)) \cap \mathit{pre}_S(\kappa)\} \end{aligned}$$

■

In terms of action physics, a disconfirm action can only be executed once, and otherwise is modelled as a self-transformation. We only consider *relevant* assumptions when constructing the abstract model. For example, taking the switching action $\tilde{a}$ to be (`look milk kitchen`) from our earlier sequential plan example, we have that $\mathcal{A}^\circ(.7; (=$ (`related-to cup`) `room0`)) is not relevant, and therefore we exclude the corresponding disconfirm action from the abstract decision process. Given $\tilde{a}$, we also include another once-only self-transition action $\mathcal{A}.\texttt{pre}(\tilde{a})$, a *confirmation action* with the reward property:

$$\mathsf{R}(s, \mathcal{A}.\texttt{pre}(\tilde{a})) = \begin{cases} \$(\texttt{pre}(\tilde{a})) & \texttt{if pre}(\tilde{a}) \subseteq s \\ \hat{\$}(\texttt{pre}(\tilde{a})) & \texttt{otherwise} \end{cases}$$

Execution of either a disconfirmation or the confirmation action exits the DT session.

Turning to the detail of (dis-)confirmation rewards, in this paper, for $\mathcal{A}^\bullet(\rho_i; T_i)$ actions we set $\$(x)$ to be a small positive constant, and have $\hat{\$}(x) = -\$(x)(1 - \rho)/\rho$ where ρ is the probability that x is true. For $\mathcal{A}.\texttt{pre}(\tilde{a})$ actions we have $\hat{\$}(x) = -\$(x)\rho/(1 - \rho)$.

In order to guarantee fast DT sessions, these plan in an abstract process determined by the current trace and underlying belief-state (Alg. 3, Line 3). The abstract process posed to the DT planner is constructed according to Algorithm 4, which first constrains as statically false all propositions except those which are true with probability 1, or which are the subject of *relevant* assumptions (Alg. 4, Line 2). The details of that constraint is formally given in terms of a *belief projection*, as follows.

Definition 2 *(Belief Projection)*

Given a set of propositions $X \subseteq \mathcal{P}$ and a belief b expressed as a DTPDDL `(:init)` *term, $\theta(X, b)$ gives a belief-state that corresponds to the interpretation of the b term where all atoms not in X are omitted.* ■

Taking our example trace with assumptive action probabilities changed to reflect the belief-state in Fig. 6B, given switching action "(look box kitchen)" the underlying belief in Fig. 6B would determine a fully constrained belief given by Fig. 6A. Next, static

Algorithm 3 DTSESSION()

```
1: Input:
     • DTPDDL problem description $\Pi$, including current belief-state $b_0$ expressed as an (:init) term
     • Sequential-plan prefix $[a_1, .., a_N, \tilde{a}]$, containing both actions from $\Pi$, and assumptive actions $\mathcal{A}^\circ$. Here, $\tilde{a}$ is the action whose scheduled execution switched planning from a sequential session (see Alg. 2) into this decision-theoretic session.
2: Initialise:
     • $\mathcal{A}^{\circ R}$, the *relevant assumptions*, see Def. 1
     • $b^a \leftarrow \texttt{FPAR}(b, \mathcal{A}^{\circ R}, \texttt{MAX})$ – see Alg. 4
     • $\mathcal{A}^{confirm} \leftarrow \{\textsc{ConfirmAction}(\tilde{a})\}$ – The confirmation actions according to Def. 1
     • $\mathcal{A}^{disconfirm} \leftarrow \{\textsc{DisconfirmAction}(a_i)\ \forall a \in \mathcal{A}^{\circ R}\}$ – The disconfirmation actions according to Def. 1
     • $\Pi^{DT} \leftarrow \Pi$ with added actions $\mathcal{A}^{confirm}$ and $\mathcal{A}^{disconfirm}$
3: $b_0^{\tilde{a}} \leftarrow \texttt{FPAR}(b_0, \mathcal{A}^{\circ R}, 200)$ – see Alg. 4
4: $a \leftarrow \text{DTI}\textsc{nit}(\Pi^{DT})$
5: while $a \notin \mathcal{A}^{confirm} \cup \mathcal{A}^{disconfirm}$ do
6:     $\Pi, o \leftarrow \textsc{ExecuteAction}(a)$
7:     $a \leftarrow \text{DTN}\textsc{ext}(\Pi^{DT}, o)$
8: end while
9: return $\Pi$
```

constraints are removed, one proposition at a time, until the number of states that can be true with non-zero probability in the initial belief of the abstract process reaches a given threshold (Alg. 4, Lines 3-8). In detail, for each statically-false proposition we compute the *entropy* of the relevant assumptions of the current trace *conditional* on that proposition, as follows.

Definition 3 *(Conjunctive Clause)*
Where a literal *ℓ is a proposition p or its negation $\neg p$, a conjunctive clause is a term of the form $\ell_1 \wedge \ell_2 \wedge .. \wedge \ell_n$, written $\bigwedge_i \ell_i$. Given a clause ϕ, we write $s \models \phi$, pronounced "s models ϕ", if each positively occurring term p in ϕ is contained in s, i.e., $p \in s$, and each negatively occurring literal $\neg p$ is not, i.e., $p \notin s$.* ■

Definition 4 *(Assignments as Conjuncts) Writing 2^X for the powerset of a set $X \subseteq \mathcal{P}$, then*

$$\chi[X] \equiv \{ \bigwedge_{p \in X' \cap X} p \wedge \bigwedge_{p \in X \setminus X'} \neg p \mid X' \in 2^X \},$$

In other words, $\chi[X]$ is a set of conjunctions each of which corresponds to one truth assignment to elements in X. ■

(A) Fully constrained belief

```
(:init (=(is-in Robot)room0)
 (.4 (and (= (category room0)kitchen)
       (.8 (= (related-to milk)room0)))
```

(B) Underlying DTPDDL belief

```
(:init (=(is-in Robot)room0)
 (.4 (and (= (category room0)kitchen)
       (.8 (= (related-to milk)room0))
       (.7 (= (related-to cup)room0)))
   .4 (and (= (category room0)office)
       (.1 (= (related-to milk)room0))
       (.5 (= (related-to cup)room0)))
       .2 (and (= (category room0)corridor)
            (.2 (= (related-to cup)room0)))))
```

(C) Partially constrained belief

```
(:init (=(is-in Robot)room0)
 (.4 (and (= (category room0)kitchen)
       (.8 (= (related-to milk)room0))
       (.7 (= (related-to cup)room0)))))
```

Figure 6: Simplified examples of belief-states from DT sessions.

Definition 5 *(Conditional Entropy) Writing $b(\phi)$ for the probability that a conjunction ϕ is satisfied in belief-state b – i.e. $b(\phi) = \sum_{s\in\mathcal{S},s\models\phi} b(s)$ – the entropy of propositions X* conditional *on a proposition y, written* $\mathrm{H}(X|y)$*, is:*

$$\mathrm{H}(X|y) = \sum_{x\in\chi[X],y'\in\{y,\neg y\}} b(x \wedge y') \log_2 \frac{b(y')}{b(x \wedge y')}$$

■

Here, a low $\mathrm{H}(X|y)$ value suggests that knowing the truth value of y is useful for determining whether or not some assumptions X hold. When removing a static constraint on propositions during construction of the abstract process (Alg. 4, Line 6), y_i is considered before y_j if $\mathrm{H}(X|y_i) < \mathrm{H}(X|y_j)$. For example, if the serial plan assumes a bottle of milk is in `room0` which we assume to be a kitchen, then those propositions are added to characterise the abstract process' states. Taking the relevant assumptions X to be $\{(=\ (\texttt{category room0})\ \texttt{kitchen}), (=\ (\texttt{is-in milk})\ \texttt{room0})\}$, in relaxing static constraints the following entropies are calculated:

```
H(X|(=(category room0) kitchen)) = 0.53
H(X|(=(category room0) office))  = 0.81
H(X|(=(category room0) corridor))= 0.59
H(X|(=(related-to milk) room0))  = 0.56
H(X|(=(related-to cup) room0))   = 0.99
```

Therefore, the first static constraint to be relaxed is for `(=(related-to cup)room0)`, giving a refined abstract belief state depicted in Fig. 6C. Summarising, if for Fig.6B the DT session is restricted to belief-states with fewer than 8 elements, then the starting belief-state of the DT session does not mention a "corridor" or "office".

6 Case-based Analysis of System Behaviour

In order to test our system and analyse the generated behaviour we ran it in a real-world office environment. To highlight its ability to achieve different tasks with one and the same system we gave the robot three different goals to achieve, in series, in

Algorithm 4 FPAR() – Factored POMDP Abstraction and Refinement

```
1: Input:
     • Current belief-state $b$ formatted as a DTPDDL (:init) term
     • Relevant assumptive actions $\mathcal{A}^{\circ R}$
     • MAX, upper limit on the number of states allowed to occur with non-zero
       probability in the initial belief of the resulting abstract process
2: Initialise:
     • $\mathcal{P}^{\top} \leftarrow \{p | b(s) > 0, p \in s\}$ – Facts that are necessarily true
     • $\mathcal{P}^{\circ} \leftarrow \{p | \mathcal{A}^{\circ}(\rho_i; T_i) \in \mathcal{A}^{\circ}, p \in T_i\}$ – Facts that are assumed true in the
       proceeding sequential session
     • $\mathcal{P}^{\perp} \leftarrow \mathcal{P} \backslash (\mathcal{P}^{\top} \cup \mathcal{P}^{\circ})$ – Facts that are statically false in a minimal abstraction
     • Abstraction index $i \leftarrow 0$
     • $\mathcal{P}_i \leftarrow \mathcal{P}^{\top} \cup \mathcal{P}^{\circ}$ – Facts that characterise states in the i'th abstraction
     • $b_i \leftarrow \theta(\mathcal{P}_i, b)$ – Belief-state of i'th abstraction, see Def. 2.
3: while Cardinality of $\{s | b_i(s) > 0\}$ is less than MAX do
4:     $i \leftarrow i + 1$
5:     Choose a $p$ from $\mathcal{P}^{\perp} \setminus \mathcal{P}_{i-1}$ so that $\mathrm{H}(\mathcal{P}^{\circ} | p)$ is minimal. See Def. 5.
6:     $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup p$
7:     $b_i \leftarrow \theta(\mathcal{P}_i, b)$, see Def. 2
8: end while
9: return $b_i$ – Starting state distribution for abstract process
```

one run. First, we tasked the robot to explore autonomously highlighting its ability to extend its map and to plan in some limited open-world. Then we gave it the goal to determine the category of one of the three rooms, and finally to find a certain object located in another room. This case-based analysis shall (i) prove the general ability to engage in a variety of different tasks and generate intelligent behaviour acquiring knowledge by invoking the right competences, (ii) serve as exemplary runs to help the understanding of the actual processing within the system including planning and reasoning, and (iii) highlight the ways our approach reasons and plans with uncertainty taken into account. In previous work [2] we were able to show that this system can successfully exploit uncertain knowledge, acquired through co-occurrence analysis of room and object categories (cf. Sec. 3.3) and through inference over room properties, to yield more efficient and robust object search behaviour. Here we build upon these findings and analyse what course of action our system chooses for a variety of different tasks.

6.1 Experimental Setup

To illustrate the capabilities of our integrated system we carried out a number of runs in a environment consisting of an office and a meeting room, linked by a corridor. The robot was given three different goals, one at a time. These goals were: to *Explore* the environment and hence build maps of interconnected places (divided into rooms) and obstacles / free space; to *Categorise* one of the rooms by asking a person if the uncertainty of the inferred categorisation (following exploration) is above a threshold; and to *Find* a specified object (a magazine) in an explored environment.

The focus for these runs was to demonstrate the ability of the system to achieve these different goals. To ensure that the initial room categorisation (produced during exploration) was sufficiently uncertain we adjusted the domain models by reducing the probability that an ‘office-like’ place was actually in an office and increased the probability that it was in a different room type (we also restricted the available room types to the three present in this environment). To show that the robot will first search in the most likely location for the magazine we adjusted the prior probabilities for its location to be high for the meeting room and low for the office and corridor. To prevent the robot from entering other offices or travelling too far down the corridor we inserted artificial barriers into the spatial subarchitecture. Only parts of the office and meeting room were accessible to the robot due to the placement of the furniture. The location of the magazine was varied between runs. User input was through GUIs rather than speech. One GUI was used to input the goals. Another GUI was used for the dialogue-based interaction for the room categorisation task. The robot generated text output which was also spoken using a text-to-speech component.

The robot started each run in the same location (the office) and orientation. The following goals were issued, in this sequence: explore; determine the category of the initial room; explore again; and find the magazine. The purpose of the second exploration goal is to ensure that the map is complete before starting the object search task, because new placeholders may be found when the robot returns to the office. We will not analyse the performance of the second exploration task.

6.2 Analysis of System Behaviour

We discuss the behaviour of the system in one of the runs. The robot’s model of the environment at the end of this run is shown in Fig. 7.

Explore!

The system was initialised in the office with the robot at Place 0 (to the left of Fig. 7) and two new placeholders were created by the mapping & navigation competence (Sec. 3.1). The robot was first given the goal of exploring the environment:

```
(forall (?p - place) (= (placestatus ?p) trueplace))
```

The exploration task is therefore carried out by planning to visit all placeholders. If the robot successfully moves to a placeholder it is converted into a Place and further placeholders are generated if new free space is discovered by the competence. This may subsequently lead to re-planning because it can result in a change of the belief state (the addition of new objects) that was not explicitly planned for. Fig. 8(a) shows the actions carried out for this goal, where the blank space between actions is due to re-planning. Each of the first 12 actions was a `move` (for its definition see line 320 in the planning domain in Appendix A) between a Place and an adjacent placeholder. This took the robot from the office, out into the corridor and down to Place 13.

From Place 13 a `move_direct` (line 336 in Appendix A) was planned to visit placeholder 15 via Place 14. As noted in Sec. 3.1, this action enables the robot to move

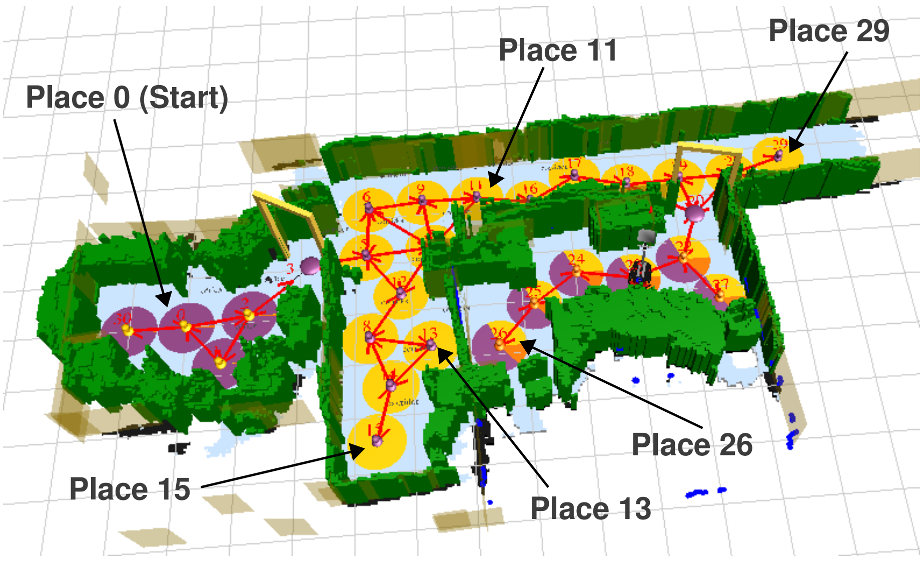


Figure 7: A visualisation of the robot's model of the environment at the end of the example run described in section 6.2. The robot is in the meeting room. The office is on the left. 3D obstacles are shown in green. Here the colours for the room categories are: purple for *office*; yellow for *corridor*; and orange for *meeting room* .

more quickly between Places because it is allowed to cut corners rather than pass exactly through the intermediate place. From Place 15 a sequence of three `move_direct` actions took the robot to Place 11 (cf Fig. 8(a)). Exploration continued down the corridor and into the meeting room, up to Place 26 (the left-most Place in the meeting room, as shown in Fig. 7). The rest of the actions in the sequence took the robot to the remaining placeholders (one in the meeting room and two in the corridor). The exploration goal was achieved when the robot reached Place 29. (Place 30, to left of Place 0, was created during the second exploration task.)

Categorise!

Next the robot was given the goal of determining the category of room 0 (the office):

```
(kval robot_0__c (category room_0_91))
```

where `robot_0__c` and `room_0_91` are the working memory addresses for the beliefs for the robot and room 0, respectively.

After the exploration goal had been achieved, the robot had the following probabilities for the possible category of this room: office 0.61; meeting room 0.35; corridor 0.04 – i.e. it was most likely an office; the robot therefore planned to verify this hypothesis by using a human as a knowledge source (cf Sec. 3.5). The robot had a prior belief that any room contains a person with probability 0.9. It therefore chose to go to room 0 to ask someone there. The actions carried out by the robot are shown in Fig. 8(b). Again, blank space is due to (re-)planning. The initial sequence of `move_direct` actions took the robot to Place 2 in room 0. After re-planning the robot executed a `look-for-people` action (see Sec. 3.6 and line 449 in Appendix A) and found that there was indeed a person in room 0 (located at Place 0 with probability 0.99). The robot then moved to Place 0 and used an `engage` (line 467 in Appendix A) action to establish a dialogue with the human. As part of this action the robot moved to Place 2 and then turned to face Place 0, before commencing the dialogue shown below:

```
robot: hello human
huan: hello dora
robot: ok
```

The planner then issued a command to move it back to Place 0. Finally, an `ask-for-category-polar` action (line 485 in Appendix A) was executed to ask the verfication question:

```
robot: is this room a office?
human: yes
robot: ok
```

The robot then updated its knowledge about the category of room 0, updating the probability of it being an office to 0.98 – hence in Fig. 7 (at the end of the run) the discs around the places in room 0 are a solid purple colour.

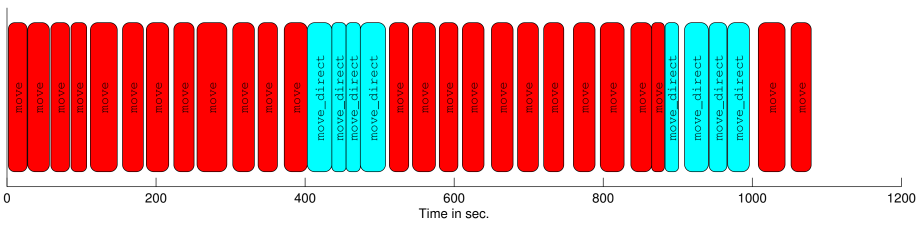
Find!

The last goal given to the robot was to find the magazine:

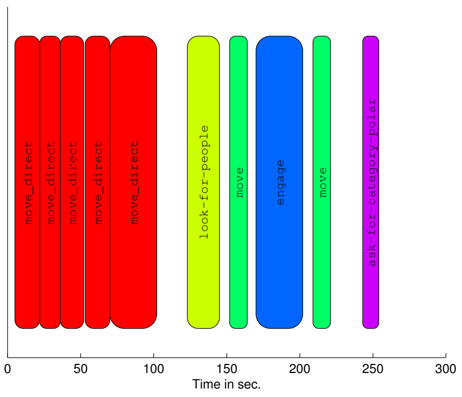
```
(exists (?o - visualobject)
        (and  (= (label ?o) magazine)
              (kval robot_0__c (related-to ?o)))
```

As noted above, the robot had a strong prior belief that the magazine is located in a meeting room. After confirming with a human that room 0 is indeed an office, the only room that might be a meeting room is room 2 (room 1 having been categorised as a corridor). The probabilities of the category of room 2 were: meeting room 0.39; office 0.39; corridor 0.22. The robot therefore planned to move to room 2 (as shown in Fig. 8(c)). Once there it executed a `create_cones_in_room` action (as described in Sec. 3.4 and defined in line 355 in Appendix A). 17 viewpoints were selected, grouped into 8 conegroups – see Fig 9 which shows the state after the robot has found the magazine (the cyan viewcones have been searched; the red ones have not). The probability of the magazine being in each of these conegroups ranged between 0.04 and 0.10, with the sum being 0.61. The planner selects one conegroup at a time to process, based on the cost of moving to the Place from the current location and the probability associated with the viewpoints in the conegroup. Fig. 8(c) shows that three `process_conegroup` actions (cf. line 422 in Appendix A) were executed, after corresponding movement actions to take the robot to the associated Place. The first conegroup processed was at Place 24 and contained two viewpoints. The second was at Place 27 (3 viewpoints) and the the third at Place 23 (2 viewpoints – the magazine was found in the second of these). Re-planning occurred after each conegroup was processed, in order to select which conegroup to process next.

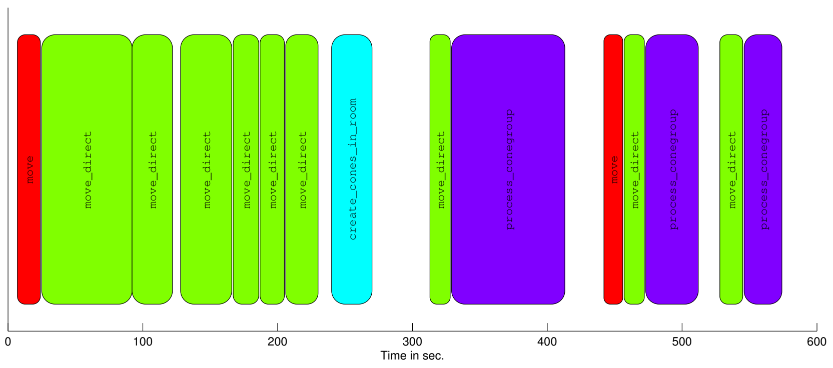
As described in Sec. 3.4, as viewpoints are processed the probability that the magazine is in the room decreases if the magazine is not detected. Therefore the probability of room 2 being a meeting room decreased from the initial value of 0.39 to 0.14 (via intermediate values of 0.28 and 0.19) as this additional information was used by the conceptual layer (cf Sec.4). This is reflected the size of the orange segment around the Places in the room in Fig. 7.



(a) Explore!



(b) Categorise!



(c) Find!

Figure 8: Sequence of actions executed for different goals, described in Sec. 6.2

7 Conclusion

We presented a mobile robot system that can accomplish a variety of epistemic goals, taking into account uncertain sensing, and exploit uncertain knowledge, by employing probabilistic representations, reasoning, and planning, to generate intelligent behaviour. The system presented has a layered architecture that features a deliberative layer accommodating planning and goal management algorithms that are completely domain-independent. This gives our system the ability to engage in a variety of tasks and achieve them in a goal-driven manner by just giving the system a different goal.

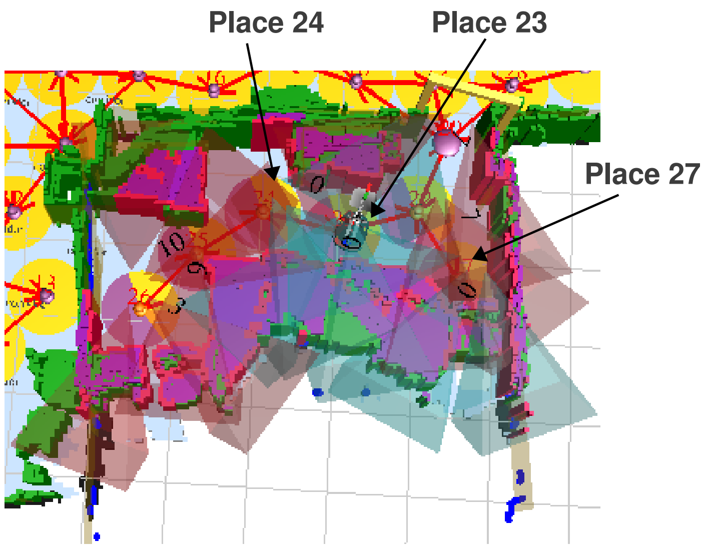


Figure 9: Viewpoints for the object search task described in Sec. 6.2. Cyan viewpoints have been searched; Red ones have not. The seven black numbers represent the probability that the magazine is contained within the viewpoints in each conegroup. The probability is reduced as viewpoints are searched; hence the zero values for the three processed conegroups.

Unique to our system is its ability to self-extend, i.e. to pursue epistemic goals to populate its belief state with new objects and additional knowledge. Our system pursues these goals by actively exploiting various knowledge sources, from using its own vision system to see the world to consulting the web for background knowledge. It can integrate the potentially unreliable evidence gathered from these sources into a coherent probabilistic representation that in turn is compiled into the belief state used to plan further actions. Key aspects of our work are: the probabilistic approach to representing both pre-defined as well as gathered information; the domain-independent planning algorithms that can plan in the vast state space resulting from the probabilistic approach; and the coherent architectural approach to implement individual competences that are controlled by the planning algorithms mediated through the conceptual layer that hosts the domain-specific models. In this paper we were able to show that with this approach our robot can autonomously explore unknown environments, it can get to know the category of different rooms, and it can find objects, all by exploiting the probabilistic domain knowledge about human-inhabited indoor environments that the robot is endowed with.

We have developed an extensible robotic system that reflects on its knowledge, detects knowledge gaps and plans for knowledge gathering actions. It contains reusable components that implement autonomous behaviour, accounting for uncertainty (in acquired as well as pre-given knowledge) and explicitly represented hypotheses. Our design allows us to integrate further competences and pursue other goals than the three classes presented in this paper by adapting the domain models in the conceptual layer, while not requiring any modification to the planning algorithms in the deliberative layer. Future work includes letting the robot choose more intelligently which goal to pursue next, extending our work on goal management [12]. Also, one of the most limiting factors of our system at present is the requirement to provide pre-trained object detectors. This limits the system to a rather small set of objects it could recognise, even though potentially we could have a large body of background knowledge about all sorts of objects. In our aim to reduce the fixed limitations of our system, merging it with the "George" robot [11] that can interactively learn the appearance of objects, following the same architectural schema and employing the same deliberative layer as we do here, is a natural candidate for a future enhancement.

References

[1] S. Dickinson, "The evolution of object categorization and the challenge of image abstraction," in *Object Categorization: Computer and Human Vision Perspectives*. Cambridge University Press, 2009, pp. 1–58. [Online]. Available: http://homepage.univie.ac.at/nicole.rossmanith/concepts/papers/dickinson2009evolution.pdf

[2] M. Hanheide, C. Gretton, R. W. Dearden, N. A. Hawes, J. L. Wyatt, A. Pronobis, A. Aydemir, M. Göbelbecker, and H. Zender, "Exploiting Probabilistic Knowledge under Uncertain Sensing for Efficient Robot Behaviour," in *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2011.

[3] H. Jacobsson, N. Hawes, G.-J. Kruijff, and J. Wyatt, "Crossmodal content binding in information-processing architectures," in *Proceedings of the 3rd international conference on Human robot interaction - HRI '08*. New York, New York, USA: ACM Press, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1349822.1349834

[4] 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 *Proc. of Int. Workshop on Hybrid Control of Autonomous Systems*, Pasadena, CA, USA, July 2009, pp. 25–32.

[5] J. L. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G.-J. M. Kruijff, P. Lison, A. Pronobis, K. Sjoo, A. Vrecko, H. Zender, M. Zillich, and D. Skocaj, "Self-Understanding and Self-Extension: A Systems and Representational Approach," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 282–303, Dec. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5613920

[6] H. I. Christensen, G.-J. M. Kruijff, and J. L. Wyatt, Eds., *Cognitive Systems*, ser. Cognitive Systems Monographs. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 8. [Online]. Available: http://www.springerlink.com/index/10.1007/978-3-642-11694-0

[7] N. Hawes and J. L. Wyatt, "Engineering intelligent information-processing systems with CAST," *Adv. Eng. Inform.*, vol. 24, no. 1, pp. 27–39, 2010.

[8] K. Sjöö, H. Zender, P. Jensfelt, G.-J. M. Kruijff, A. Pronobis, N. Hawes, and M. Brenner, "The Explorer system," in *Cognitive Systems*, H. I. Christensen, G.-J. M. Kruijff, and J. L. Wyatt, Eds. Springer, April 2010, pp. 395–421. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11694-0_10

[9] N. Hawes, M. Hanheide, J. Hargreaves, B. Page, and H. Zender, "Home alone: Autonomous extension and correction of spatial representations," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, 2011, to appear.

[10] S. Edelkamp and P. Kissmann, “Pddl 2.1: The language for the classical part of ipc-4,” in *Proceedings of the International Planning Competition. International Conference on Automated Planning and Scheduling. Whistler, Canada*, 2004.

[11] D. Skočaj, M. Kristan, A. Vrečko, M. Mahnič, M. Janíček, G.-J. M. Kruijff, M. Hanheide, N. Hawes, T. Keller, M. Zillich, and K. Zhou, “A system for interactive learning in dialogue with a tutor,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2011*, San Francisco, CA, USA, 25-30 September 2011. [Online]. Available: http://cogx.eu/data/cogx/publications/skocajIROS11.pdf

[12] M. Hanheide, N. Hawes, J. L. Wyatt, M. Göbelbecker, M. Brenner, K. Sjöö, A. Aydemir, P. Jensfelt, H. Zender, and G.-J. M. Kruijff, “A Framework for Goal Generation and Management,” in *Proceedings of the AAAI Workshop on Goal-Directed Autonomy*, 2010.

[13] A. Pronobis, K. Sjöö, A. Aydemir, A. N. Bishop, and P. Jensfelt, “A framework for robust cognitive spatial mapping,” in *Proceedings of the 14th International Conference on Advanced Robotics (ICAR’09)*, Munich, Germany, June 2009.

[14] H. Zender, O. M. Mozos, P. Jensfelt, G.-J. M. Kruijff, and W. Burgard, “Conceptual spatial representations for indoor mobile robots,” *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 493–502, June 2008.

[15] N. Hawes, M. Hanheide, J. Hargreaves, B. Page, H. Zender, and P. Jensfelt, “Home alone: Autonomous extension and correction of spatial representations,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011, pp. 3907–3914. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5980004

[16] A. Pronobis and P. Jensfelt, “Large-scale semantic mapping and reasoning with heterogeneous modalities,” in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA’12)*, Saint Paul, MN, USA, May 2012.

[17] A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt, “Multi-modal semantic place classification,” *Int. J. Robot. Res.*, vol. 29, no. 2-3, pp. 298–320, February 2010.

[18] T. Kollar and N. Roy, “Utilizing object-object and object-scene context when planning to find things,” in *ICRA’09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 4116–4121.

[19] 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’11)*, Örebro, Sweden, Sept. 2011.

[20] S. Ekvall, D. Kragic, and P. Jensfelt, “Object detection and mapping for service robot tasks,” *Robotica: International Journal of Information, Education and Research in Robotics and Artificial Intelligence*, 2007.

[21] H. González-Banos, "A randomized art-gallery algorithm for sensor placement," in *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry*. New York, NY, USA: ACM, 2001, pp. 232–240.

[22] K. Sjöö, A. Aydemir, D. Schlyter, and P. Jensfelt, "Topological spatial relations for active visual search," Centre for Autonomous Systems, KTH, Stockholm, Tech. Rep. TRITA-CSC-CV 2010:2 CVAP317, July 2010.

[23] A. Aydemir, K. Sjöö, J. Folkesson, and P. Jensfelt, "Search in the real world: Active visual object search based on spatial relations," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[24] A. Richtsfeld, T. Mörwald, M. Zillich, and M. Vincze, "Taking in shape: Detection and tracking of basic 3d shapes in a robotics context," in *Computer Vision Winder Workshop*, 2010, pp. 91–98.

[25] 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)*, Pasadena, CA, USA, July 2009, pp. 1604–1609.

[26] M. Schröder and J. Trouvain, "The German text-to-speech synthesis system MARY: A tool for research, development and teaching," *International Journal of Speech Technology*, no. 6, pp. 365–377, 2003.

[27] R. H. Thomason, M. Stone, and D. DeVault, "Enlightened update: A computational architecture for presupposition and other pragmatic phenomena," in *Presupposition Accommodation*, D. Byron, C. Roberts, and S. Schwenter, Eds. Ohio State Pragmatics Initiative, 2006.

[28] M. Brenner and B. Nebel, "Continual planning and acting in dynamic multiagent environments," *Autonomous Agents and Multi-Agent Systems*, vol. 19, pp. 297–331, 2009, 10.1007/s10458-009-9081-1. [Online]. Available: http://dx.doi.org/10.1007/s10458-009-9081-1

[29] M. Janíček, "Abductive reasoning for continual dialogue understanding," in *New Directions in Logic, Language, and Computation*, M. Slavkovik and D. Lassiter, Eds. Springer, 2012.

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

[31] M. Schröder, E. Bevacqua, R. Cowie, F. Eyben, H. Gunes, D. Heylen, M. ter Maat, G. McKeown, S. Pammi, M. Pantic, C. Pelachaud, B. Schuller, E. de Sevin, M. Valstar, and M. Wöllmer, "Building autonomous sensitive artificial listeners," *IEEE Transactions on Affective Computing*, vol. 99, no. 1, 2011.

[32] S. C. Pammi, M. Schröder, M. Charfuelan, O. Türk, and I. Steiner, "Synthesis of listener vocalisations with imposed intonation contours," in *Seventh ISCA Tutorial and Research Workshop on Speech Synthesis*. ISCA, 2010.

[33] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings International Conference on Image Processing*, vol. 1, no. 1, IEEE. Ieee, 2002, pp. 900–903. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1038171

[34] A. Pronobis, K. Sjöö, A. Aydemir, A. N. Bishop, and P. Jensfelt, "Representing spatial knowledge in mobile cognitive systems," in *11th International Conference on Intelligent Autonomous Systems (IAS-11)*, Ottawa, Canada, Aug. 2010.

[35] J. M. Mooij, "libDAI: A free and open source C++ library for discrete approximate inference in graphical models," *J. Mach. Learn. Res.*, vol. 11, pp. 2169–2173, Aug. 2010. [Online]. Available: http://www.jmlr.org/papers/volume11/mooij10a/mooij10a.pdf

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

[37] D. Bryce, S. Kambhampati, and D. E. Smith, "Sequential monte carlo in reachability heuristics for probabilistic pl anning," *Artif. Intell.*, vol. 172, pp. 685–715, April 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1342435.1342789

[38] S. Yoon, A. Fern, R. Givan, and S. Kambhampati, "Probabilistic planning via determinization in hindsight," in *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*. AAAI Press, 2008, pp. 1010–1016. [Online]. Available: http://portal.acm.org/citation.cfm?id=1620163.1620229

A Planning Domain

```
(define (domain dora)
  (:requirements :mapl :adl :fluents :partial-observability :dynamic-objects
      :action-costs)

  (:types
   conegroup - object
   robot - planning_agent
   person robot - location
   label category spatial_relation place room visualobject - object
   visualobject room - location
   place_status - object
   )

  (:predicates
   (connected ?p1 ?p2 - place)
   (engaged ?p - person)
   (is-visited ?c - conegroup)


   (is-virtual ?o - object)
   (position-reported ?o - visualobject)

   ;; derived predicates
   (attached_to_room ?p - place ?r - room)
   (cones_exist  ?l - label ?rel - spatial_relation ?where - (either visualobject
        room))

   ;;virtual predicates
   (cones_created  ?l - label ?rel - spatial_relation ?where - (either
        visualobject room))
   )


  (:functions
   (is-in ?o - robot) - place
   (is-in ?o - person) - place
```

```

    ;; === Default knowledge ===

    ;; expected cost of searching for an object. Used by CP planner
    (dora__cost_inroom ?l - label) - number
    (dora__cost_inobject ?l1 ?l2 - label) - number
    (dora__cost_on ?l1 ?l2 - label) - number
    (search_cost ?l - label ?rel - spatial_relation ?where - (either visualobject
         room)) - number
    ;; default probabilities. These come from Coma.
    (dora__inroom ?l - label ?c - category) - number
    (dora__inobject ?l1 ?l2 - label ?c - category) - number
    (dora__on ?l1 ?l2 - label ?c - category) - number

    ;; === inferred knowledge ===

    ;; The result of applying the default knowledge.

    ;; E.g. category(r1) = kitchen AND (dora__in_room cornflakes
    ;; kitchen) => (obj_exists cornflakes in kitchen)
    ;; Also see the rules below
    (obj_exists ?l - label ?rel - spatial_relation  ?where - (either visualobject
         room)) - boolean
    (p-obj_exists ?l - label ?rel - spatial_relation  ?where - (either
         visualobject room) ?c - category) - number

    ;; === room properties ===
    (category ?r - room) - category
    (identity ?r - room) - category
    (roomid ?r - room) - number
    (virtual-category ?r - room) - category
    (virtual-place ?r - room) - place

    ;; === place properties ===
    (placestatus ?n - place) - place_status
    (in-room ?p - place) - room
    (place-exists ?p - place) - boolean

    ;; === placeholder properties ===
    (leads_to_room ?p - place ?c - category) - boolean

    ;; === person properties ===
    (associated-with ?p - person) - room
    (does-exist ?p - person) - boolean
    (contains-a-person-prior ?r - room) - boolean

    (unresponsive ?p - person) - boolean

    ;; === object properties ===
    (label ?o - visualobject) - label
    (related-to ?o - visualobject) - location
    (relation ?o - visualobject) -  spatial_relation

    ;; === conegroup properties ===
    ;; basic properties that determine what the conegroup was generated
    ;; for (e.g. cone group for cornflakes ON table_1)
    (cg-label ?c - conegroup) - label
    (cg-relation ?c - conegroup) - spatial_relation
    (cg-related-to ?c - conegroup) - (either visualobject room)
    (cg-place ?c - conegroup) - place
    ;; probability of seeing an object of type (label ?c) when looking
    (p-visible ?c - conegroup) - number
    ;; The ground truth. Distribution should conform to the probability
    ;; above.  Assumes that an object can only be seen from one CG
    (visible_from ?o - visualobject) - conegroup


    ;; === explanation properties ===
    (entity-exists ?o - object) - boolean

    )

   (:constants
    placeholder trueplace - place_status
```

```
 in on - spatial_relation)

;;
;; Rules that are applied to the initial state. Used to create
;; virtual objects.
;;

;; create virtual visualobjects for each label
(:init-rule objects
            :parameters(?l - label)
            :precondition (not (exists (?o - visualobject)
                                       (and (= (label ?o) ?l)
                                            (is-virtual ?o))))
            :effect (create (?o - visualobject)
                            (and (is-virtual ?o)
                                 (assign (label ?o) ?l)
                                 (assign (related-to ?o) UNKNOWN)))
            )

;; create virtual persons for each (nonvirtual) room
(:init-rule persons
            :parameters(?r - room)
            :precondition (and (not (is-virtual ?r))
                               (not (exists (?p - person ?pl - place)
                                       (or (and (= (in-room ?pl) ?r)
                                                (in-domain (is-in ?p) ?pl))
                                            (and (is-virtual ?p)
                                                 (= (associated-with ?p)
                                                     ?r))))))
            :effect (create (?p - person) (and
                                           (is-virtual ?p)
                                           (assign (associated-with ?p) ?r)))
            )

;; create virtual rooms for each placeholder
(:init-rule rooms
            :parameters(?p - place)
            :precondition (and (= (placestatus ?p) placeholder)
                               (not (exists (?r - room)
                                            (and (= (virtual-place ?r) ?p)
                                                 (is-virtual ?r)))))
            :effect (create (?r - room)
                            (and (is-virtual ?r)
                                 (assign (virtual-place ?r) ?p)))
            )


;; Set undefined search costs to default values

(:init-rule default_search_costs_for_room
            :parameters (?l - label  ?r - room)
            :precondition (= (search_cost ?l in ?r) unknown)
            :effect (assign (search_cost ?l in ?r) (dora__cost_inroom ?l))
            )

(:init-rule default_search_costs_for_object_in
            :parameters (?l - label ?o - visualobject)
            :precondition (= (search_cost ?l in ?o) unknown)
            :effect (assign (search_cost ?l in ?o) (dora__cost_inobject ?l
                 (label ?o)))
            )

(:init-rule default_search_costs_for_object_on
            :parameters (?l - label ?o - visualobject)
            :precondition (= (search_cost ?l on ?o) unknown)
            :effect (assign (search_cost ?l on ?o) (dora__cost_on ?l (label
                 ?o)))
            )



;;
;; Axioms
;;

```

```
;; Used to model whether a room is fully explored

(:derived (attached_to_room ?p - place ?r - room)
          (exists (?p2 - place) (and (= (in-room ?p2) ?r)
                                     (not (= (placestatus ?p) trueplace))
                                     (connected ?p2 ?p))))

(:derived (not_fully_explored ?r - room)
          (exists (?p - place) (and (attached_to_room ?p ?r))))


(:derived (cones_exist ?l - label ?rel - spatial_relation ?where - (either
     visualobject room))
          (exists (?c - conegroup) (and (= (cg-label ?c) ?l)
                                        (= (cg-related-to ?c) ?where)
                                        (= (cg-relation ?c) ?rel))))



;;
;; Assumptions
;;


;; rules that model the conditional probabilities from default.sa

;; p(?label IN ?room | category(?room) = ?cat)
(:assume obj_in_room
         :parameters (?l - label ?r - room ?c - category)
         :precondition (and (= (category ?r) ?c)
                            (defined (dora__inroom ?l ?c))
                            (not (defined (p-obj_exists ?l in ?r ?c))))
         :effect (probabilistic (dora__inroom ?l ?c) (assign (obj_exists ?l in
              ?r) true)))


;; p(?label IN ?object | label(?object) = ?l2 AND ?object IN ?room
;;                                        AND category(?room) = ?cat)
(:assume obj_in_obj
         :parameters (?l1 ?l2 - label ?o - visualobject ?r - room ?c - category)
         :precondition (and (= (category ?r) ?c)
                            (= (label ?o) ?l2)
                            (= (related-to ?o) ?r)
                            (= (relation ?o) in)
                            (defined (dora__inobject ?l1 ?l2 ?c))
                            (not (defined (p-obj_exists ?l1 in ?o ?c))))
         :effect (probabilistic (dora__inobject ?l1 ?l2 ?c) (assign (obj_exists
              ?l1 in ?o) true)))


;; p(?label ON ?object | label(?object) = ?l2 AND ?object IN ?room
;;                                        AND category(?room) = ?cat)
(:assume obj_on_obj
         :parameters (?l1 ?l2 - label ?o - visualobject ?r - room ?c - category)
         :precondition (and (= (category ?r) ?c)
                            (= (label ?o) ?l2)
                            (= (related-to ?o) ?r)
                            (= (relation ?o) in)
                            (defined (dora__on ?l1 ?l2 ?c))
                            (not (defined (p-obj_exists ?l1 on ?o ?c))))
         :effect (probabilistic (dora__on ?l1 ?l2 ?c) (assign (obj_exists ?l1
              on ?o) true)))


;; use posterior information from conceptual.sa
;; force commitment to a room category to help the heuristic
(:assume object_existence_room
         :parameters (?l - label ?rel - spatial_relation ?where - room ?c -
              category)
         :precondition (and (= (category ?where) ?c)
                            (defined (p-obj_exists ?l ?rel ?where ?c)))
         :effect (probabilistic (p-obj_exists ?l ?rel ?where ?c) (and (assign
              (obj_exists ?l ?rel ?where) true)))
         )

```

```
;; use posterior information from conceptual.sa
(:assume object_existence_object
        :parameters (?l - label ?rel - spatial_relation ?where - visualobject
             ?r - room ?c - category)
        :precondition (and (= (related-to ?where) ?r)
                           (= (category ?r) ?c)
                           (defined (p-obj_exists ?l ?rel ?where ?c)))
        :effect (probabilistic (p-obj_exists ?l ?rel ?where ?c) (and (assign
             (obj_exists ?l ?rel ?where) true)))
        )

;; p(?label IN ?room | category(?room) = ?cat)
(:assume person_in_room
        :parameters (?p - person ?pl - place ?r - room)
        :precondition (and (= (contains-a-person-prior ?r) true)
                           (= (in-room ?pl) ?r)
                           (= (associated-with ?p) ?r)
                           (is-virtual ?p))
        :effect (probabilistic 1.0 (assign (is-in ?p) ?pl))) ;; will
             automatically be normalised

;; probability of an object being at a specific location
;; used only by DT
(:assume sample_object_location
        :parameters (?o - visualobject ?l - label ?rel - spatial_relation
             ?where - (either visualobject room))
        :precondition (and (= (label ?o) ?l)
                           (is-virtual ?o)
                           (= (obj_exists ?l ?rel ?where) true))
        :effect (probabilistic 1.0 (and (assign (related-to ?o) ?where)
                                         (assign (relation ?o) ?rel)))
        )

;; probability of finding a specific object in a conegroup
;; used only by DT
(:assume sample_cone_visibility
        :parameters (?o - visualobject ?c - conegroup ?l - label ?rel -
             spatial_relation ?where - (either visualobject room))
        :precondition (and (= (cg-relation ?c) ?rel)
                           (= (cg-related-to ?c) ?where)
                           (= (relation ?o) ?rel)
                           (= (related-to ?o) ?where)
                           (= (cg-label ?c) ?l)
                           (= (label ?o) ?l)
                           (not (is-visited ?c)))
        :effect (probabilistic (p-visible ?c) (assign (visible_from ?o) ?c))
        )


;; Assign virtual room to a placeholder
(:assume room_from_placeholder
        :parameters (?p - place ?r - room ?c - category)
        :precondition (and (= (placestatus ?p) placeholder)
                           (= (virtual-place ?r) ?p)
                           (= (leads_to_room ?p ?c) true)
                           (is-virtual ?r))
        :effect (and (probabilistic 1.0 (and (assign (in-room ?p) ?r)
                                              (assign (category ?r) ?c)))
                     (increase (total-cost) 10))
        )

;;
;; Actions
;;

;; Report the position of a found object to a person
;; precondition: robot has found the object
;;              is engaged to the person and at the same place
(:action report_position
        :agent (?a - robot)
        :parameters (?o - visualobject)
        :variables (?p - place ?h - person)
        :precondition (and (kval ?a (related-to ?o))
                           (engaged ?h)
```

```
                              (= (is-in ?h) ?p)
                              (= (is-in ?a) ?p))
          :effect (and (position-reported ?o)
                        (increase (total-cost) 1))
          )


  ;; Moves from one place to another connected place
  ;; precondition: places must be connected
  ;;               robot must be at the start place
  (:action move
          :agent (?a - robot)
          :parameters (?to - place)
          :variables (?from - place)
          :precondition (and (or (connected ?from ?to)
                                 (connected ?to ?from))
                             (= (is-in ?a) ?from))
          :effect (and (assign (is-in ?a) ?to)
                        (assign (placestatus ?to) trueplace)
                        (kval ?a (in-room ?to))
                        (increase (total-cost) 2))
          )

  ;; Moves from one place to another via a third one (cutting corners)
  ;; precondition: places must be connected
  ;;               robot must be at the start place
  (:action move_direct
          :agent (?a - robot)
          :parameters (?to - place)
          :variables (?from - place ?via - place)
          :precondition (and (or (connected ?from ?via)
                                 (connected ?via ?from))
                             (or (connected ?via ?to)
                                 (connected ?to ?via))
                             (= (is-in ?a) ?from))
          :effect (and (assign (is-in ?a) ?to)
                        (assign (placestatus ?to) trueplace)
                        (kval ?a (in-room ?to))
                        (increase (total-cost) 3))
          )


  ;; create cones for search in a room
  ;; precondition: robot is in the specified room
  (:action create_cones_in_room
          :agent (?a - robot)
          :parameters (?l - label ?r - room)
          :variables (?p - place)
          :precondition (and (= (is-in ?a) ?p)
                             (= (in-room ?p) ?r)
                             (poss (obj_exists ?l in ?r) true)
                             (not (not_fully_explored ?r)))
          :effect (and (cones_created ?l in ?r)
                        (increase (total-cost) 5))
          )

  ;; create cones for search in or on another object
  ;; precondition: robot is in the same room as the specified object
  (:action create_cones_at_object
          :agent (?a - robot)
          :parameters (?l ?lsupp - label ?rel - spatial_relation ?o -
              visualobject ?r - room)
          :variables (?p - place)
          :precondition (and (= (is-in ?a) ?p)
                             (= (label ?o) ?lsupp)
                             (poss (obj_exists ?l ?rel ?o) true)
                             (= (in-room ?p) ?r)
                             (= (related-to ?o) ?r))
          :effect (and (cones_created ?l ?rel ?o)
                        (increase (total-cost) 4))
          )

  ;; Abstract search action for the CP planner
  ;; Searches for an object in the room
```

```
;; precondition: robot is in the specified room
(:action search_for_object_in_room
        :agent (?a - robot)
        :parameters (?l - label ?r - room)
        :variables (?p - place ?o - visualobject)
        :precondition (and (= (is-in ?a) ?p)
                           (= (in-room ?p) ?r)
                           (= (label ?o) ?l)
                           (or (cones_created ?l in ?r)
                               (cones_exist ?l in ?r))
                           (poss (related-to ?o) ?r)
                           (poss (relation ?o) in))
        :effect (and (increase (total-cost) (search_cost ?l in ?r)))
        :sense (= (related-to ?o) ?r)
        )

;; Abstract search action for the CP planner
;; Searches for an object ON another object
;; precondition: robot is in the same room as the specified object
(:action search_for_object_at_object
        :agent (?a - robot)
        :parameters (?l - label ?o - visualobject ?rel - spatial_relation)
        :variables (?p - place ?r - room ?o2 - visualobject)
        :precondition (and (= (is-in ?a) ?p)
                           (= (in-room ?p) ?r)
                           (= (related-to ?o) ?r)
                           (= (label ?o2) ?l)
                           (or (cones_created ?l ?rel ?o)
                               (cones_exist ?l ?rel ?o))
                           (poss (related-to ?o2) ?o)
                           (poss (relation ?o2) ?rel))
        :effect (and (increase (total-cost) (search_cost ?l ?rel ?o)))
        :sense (related-to ?o2)
        )


;; process one conegroup
;; precondition: robot is at the location of the conegroup
(:action process_conegroup
        :agent (?a - robot)
        :parameters (?c - conegroup)
        :variables (?p - place)
        :precondition (and (= (cg-place ?c) ?p)
                           (= (is-in ?a) ?p))
        :effect (increase (total-cost) 15)
        )


(:observe visual_object
         :agent (?a - robot)
         :parameters (?c - conegroup ?o - visualobject ?l - label ?where -
              (either visualobject room) ?p - place)
         :execution (process_conegroup ?a ?c ?p)
         :precondition (and (= (label ?o) ?l)
                            (= (cg-label ?c) ?l)
                            (= (cg-related-to ?c) ?where))

         :effect (and (when (= (visible_from ?o) ?c)
                        (observed (related-to ?o) ?where)))
         )


;; Run the person detector at a place
;; precondition: Robot must be at that place
(:action look-for-people
        :agent (?a - robot)
        :variables (?p - place)
        :precondition (= (is-in ?a) ?p)
        :effect (and
                (increase (total-cost) 10))
        )

(:observe person
         :agent (?a - robot)
```

```
            :parameters (?p - person ?pl - place)
            :execution (look-for-people ?a ?pl)
            :effect (and (when (= (is-in ?p) ?pl)
                           (probabilistic 0.7 (observed (does-exist ?p) true)))
                         (when (not (= (is-in ?p) ?pl))
                           (probabilistic 0.001 (observed (does-exist ?p)
                               true))))
            )

   ;; Engage with a person
   ;; precondition: Robot must be at the location of the person
   ;;               The person must not be known to be unresponsive
   (:action engage
           :agent (?a - robot)
           :parameters (?h - person)
           :variables (?p - place)
           :precondition (and (not (= (unresponsive ?h) true))
                              (= (is-in ?h) ?p)
                              (= (is-in ?a) ?p))
           :effect (and (engaged ?h)
                        (increase (total-cost) 1)
                        (assign (failure-cost) 50))
           )

   ;; Ask a person whether the current room is of a particular category
   ;; precondition: Robot must be at the location of the person
   ;;               Robot must be in the room it asks about
   (:action ask-for-category-polar
           :agent (?a - robot)
           :parameters (?r - room ?c - category)
           :variables (?h - person ?p - place)
           :precondition (and (engaged ?h)
                              (= (is-in ?h) ?p)
                              (= (is-in ?a) ?p)
                              (= (in-room ?p) ?r))
           :effect (increase (total-cost) 5)
           )

   (:observe room-category
            :agent (?a - robot)
            :parameters (?h - person ?c - category ?p - place ?r - room)
            :execution (ask-for-category-polar ?a ?r ?c ?h ?p)
            :precondition (and
                           (engaged ?h)
                           (= (is-in ?h) ?p))

            :effect (and (when (= (category ?r) ?c)
                           (probabilistic 0.95 (observed (identity ?r) ?c))))
            )

)
```

Explaining Surprises During Continual Planning*

Moritz Göbelbecker
Albert-Ludwigs-Universität
Freiburg, Germany

May 30, 2012

Abstract

Continual planning is an effective approach to decision making in uncertain dynamic worlds. It involves creating plans based on assumptions about the real world and replanning if those plans fail. We discuss methods for making these assumptions explicit and providing explanations why a continual planning task may have failed or produced unexpected outcomes.

1 Introduction

Every plan created for a real-world system contains assumptions – and its execution will succeed if these assumptions apply to the current situation. Conversely, if the execution of a plan fails, there is at least one assumption that was incorrect. It may be desirable to find out what these assumptions were for several reasons: we may want to inform a human operator why a tasks did not succeed (possibly giving them the chance to improve the domain model), or we may try to use the *explanations* we found to improve the system behaviour for future tasks.

Assumptions can have a number of sources: Reasoning about – or even creating – a full model of the agent and its environment is usually not feasible for even moderately complex systems. Every model that is used for planning will invariably be an abstraction of the real system. When employing *continual planning*, the planner creates optimistic plans and executes those plans as long as they remain viable. These plans usually make assumptions that some unobservable or uncertain state variables have certain values that would make the plan work, or that some action with probabilistic effects will have exactly the outcome we desire.

We can roughly divide assumptions into two categories: *Implicit assumptions* are those that come from the way the domain is modelled, choices made for discretisations of continuous variables, preconditions and effects that were left out either because the domain designer did not deem them relevant, did not consider them or because they would have a too large impact on the complexity of the resulting planning problem

*The research reported here was performed in the EU FP7 IP "CogX: Cognitive Systems that Self-Understand and Self-Extend" (ICT-215181).

(either for the planner or for the designer). The following is a model of a *move* action for a robotic system:

```
(:action move
         :parameters (?r - robot ?from ?to - place)
         :precondition (and (= (connected ?from ?to) true)
                            (= (is-in ?r) ?from))
         :effect (and (assign (is-in ?a) ?to)))
```

It contains several implicit assumptions: That the robot is not stuck, that the path between the places is not blocked, that the robot's battery is charged and many more that this author did not think of.

In our continual planning framework, we introduce *explicit assumptions* to deal with uncertain states. They are special actions that the planner may apply to the initial state which make some previously uncertain facts true. As these assumptions can have costs which are higher for more unlikely assumptions, this leads the planner to prefer not only short plans but also more probable ones. For the "move" example, the truth value of a `connected` variable may be uncertain, requiring the planner to explicitly plan in an appropriate assumption (and incurring its costs) before it can use the move action.

We note that continual planning does not require the use of explicit assumptions. If we were to replace, for example, the precondition by

```
(and (not (= (connected ?from ?to) false))
          (= (is-in ?r) ?from))
```

the planner would implicitly assume a MOVE action to be applicable, unless the connectivity is known to be false. In our system, the `connectivity` variable must either be set in the initial state or made true by an appropriate assumption.

If during the continual planning process an action does not have the desired outcome, we must conclude that some assumption did not hold. If the robot is still at its original position after executing a move action, the two places may not have been connected, they may be blocked, or the battery is not charged. Our goal is to find a consistent and likely explanation for the observations we made during action execution. For example, the explanation that the battery is not charged would not be consistent with a later move action succeeding. On the other hand, several move actions failing in the same way would make that explanation more likely.

In this paper we deal exclusively with explaining surprises that result from *explicit* assumptions being violated. As it is sometimes useful to work with implicit assumptions for normal planning, we will also describe how to make assumptions explicit only for the purpose of finding explanations.

In the next section, we will describe our continual planning framework and formalise the concept of a *continual planning task*. We then describe how finding explanations for surprises in these tasks can be formulated as a planning problem, and how *relevant* explanations can be identified. We finish with a overview on related work and conclude.

2 Framework

Our continual planner is based on a series of SAS$^+$planning tasks [Bäckström and Nebel, 1995]. A SAS$^+$task Π is a tuple $\langle \mathcal{V}, \mathcal{A}, s_0, g \rangle$ of actions, variables, an initial state and a goal description. Each variable $x \in variables$ has a finite *domain* $\mathcal{D}^x$ with the *unknown* value $\perp \in \mathcal{D}^x \ \forall x \in \mathcal{V}$. A state s can be regarded as a set of facts $v = x$ which assigns to each $v \in \mathcal{V}$ an element of its domain. By default, every variable is set to the unknown value, we use $\mathrm{def}(s) = \{x = v \in s : v \neq \perp\}$ to denote the set of *defined* variables of a state. The set of actions $\mathcal{A}$ can be further divided into *physical actions* $\mathcal{A}^p$ and *assumptive actions* $\mathcal{A}^a$. An action $a \in \mathcal{A}$ has a set of preconditions, $\mathrm{pre}(a)$, a set of (possibly conditional effects) $\mathrm{eff}(a)$ and associated costs $\mathrm{c}(a)$.

Definition 1 *An assumptive action $a \in \mathcal{A}^a$ is action with the following restrictions:*

- *In every plan π, a must occur before any physical action $a^p \in \mathcal{A}^p$.*
- *All effects in $\mathrm{eff}(a)$ are of the form $x = v, v \neq \perp$.*
- *If a has an effect $x = v$, $x = \perp$ must be part of the precondition.*

Assumptions can be used to model probabilistic initial states. If a variable x has a discrete value distribution P with $P(v_0) = p_0, \ldots, P(v_n) = p_n$, we can create a set of assumptions $a_0, \ldots, a_n$ that represent this distribution by setting $\mathrm{pre}(a_i) = \{x = \perp\}$, $\mathrm{eff}(a_i) = \{x = v_i\}$ and $\mathrm{c}(a_i) = -\log(p_i)$. Conditional probabilities $P(X|Y)$ can be modelled in a similar way by including preconditions for y.

In general every probability distribution can be modelled by assumptions, though in extreme cases this may require enumerating the joint distribution of all uncertain variables. We concentrate on distributions that can be represented (or approximated by) using a factored representation, s.t. every assumption a has only one effect of the form $X = v$.

Definition 2 (Action applicability) *A physical action a is* applicable *in state s, iff* $\mathrm{pre}(a) \subseteq s$.

An assumption a is applicable in s iff the following holds for any precondition $x = v \in \mathrm{pre}(a)$*:*

- $x = v \in s$
- $v = \perp$ *and* $\exists v' \in \mathcal{V} : x = v' \in eff(a) \land x = v' \in s$

The latter conditions makes sure that assumptions that are (partially) confirmed remain applicable. For reasons of brevity, we refrain from giving a formal definition from action and plan application, as those use normal SAS$^+$semantics. We refer to the result of applying action a in state s as $\mathrm{app}(s, a)$ and of applying a sequence of actions $a_0, \ldots, a_n$ as $\mathrm{app}(s, a_0, \ldots, a_n)$.

Definition 3 (Plan validity) *A sequence of actions $a_0, \ldots, a_n$ is valid given a goal g in state s if the following holds:*

- a_0 *is applicable in* s
- a_i *is applicable in* $\text{app}(s, a_0, \ldots, a_{i-1})\ \forall 1 \leq i \leq n$.
- $\text{app}(s, a_0, \ldots, a_n)$ *satisfies* g.

Using those definitions, Algorithm 1, describes the continual planning process, referring to some variables that we will use later for analysing the process. It differs from most representations of continual planning by the special treatment of assumptive actions and by keeping track of executed and unexecuted actions and intermediate states. π_i^e refers to the *executed* actions of the i-th plan and is initialised in line 13. Every time an action is executed, it is removed from the unexecuted plan π_i^u (line 16) and appended to π_i^e (line 18). As assumptive actions are not executable, they are split from the set of unexecuted actions in line 13.

Algorithm 1 Continual Planning

```
1:  Input: initial state $s_0$, goal $g$
2:  Output: success or failure
3:  $i \leftarrow 0$
4:  loop
5:      if CheckGoal($s_i$) then
6:          return success
7:      end if
8:      $\pi_i \leftarrow$ CreatePlan($s_i, g$)
9:      if $\pi_i = \emptyset$ then
10:         return failure
11:     end if
12:     $\pi_i^e \leftarrow []$
13:     $\pi_i^a, \pi_i^u \leftarrow \pi_i$
14:     $s_{i+1} \leftarrow s_i$
15:     while $\pi_i^a, \pi_i^u$ is valid in $s_{i+1}$ do
16:         $a, \pi_i^u \leftarrow \pi_i^u$
17:         $s_{i+1} \leftarrow$ Execute($a$)
18:         $\pi_i^e \leftarrow \pi_i^e, a$
19:     end while
20:     $i \leftarrow i + 1$
21: end loop
```

Using the definitions from Algorithm 1, we can define a complete continual planning process as follows:

Definition 4 *A* continual planning process $\mathcal{P}_\Pi = \langle \Pi, \mathcal{S}_\pi, \mathcal{S}_s \rangle$ *consists of a underlying planning problem* Π*, a sequence of plans* $\mathcal{S}_\pi = [\pi_0, \ldots, \pi_n]$ *and a sequence of observed states* $\mathcal{S}_s = [s_0, \ldots, s_{n+1}]$*, with* s_0 *being equal to the initial state of the planning problem. Each plan* $\pi_i = \pi_i^a, \pi_i^e, \pi_i^u$ *can be divided into assumptions, executed plan and unexecuted plan and we refer to* $\text{app}(s_i, \pi_i^a, \pi_i^e) = s_i'$ *as the* expected state *after execution.*

3 Surprises, Failures and Explanations

Given the formal definition of a continual planning process, we can now proceed with defining what precisely we mean by a "surprise" and a "failure". Our notion of surprise is based on the differences between the expected states s'_i and the perceived states s_{i+1}.

Definition 5 *Let $\mathcal{P}_\Pi$ be a continual planning process, then the* surprises *$\mathcal{S}$ of $\mathcal{P}_\Pi$ are a sequence of subsets of the differences between expected and observed states:*

$$\begin{aligned} \mathcal{S} &= [\mathcal{S}_0, \ldots, \mathcal{S}_n] \\ \mathcal{S}_i &\subseteq s_{i+1} \setminus s'_i \end{aligned}$$

This definition of "surprise" is quite general: any observed fact that was not predicted by the domain model can be considered as a failure. This may be a too broad definition for many use cases, but the analyses of the general case also work for restricted cases. A useful subset of surprises is what we call *failures*: it consists only of those unpredicted facts that cause the rest of the plan to become invalid.

Definition 6 *Given a sequence of surprises $\mathcal{S}$, an* explanation *$\mathcal{E}$ for $\mathcal{S}_i$ is a sequence of assumptive actions so that:*

$$\begin{aligned} &\mathcal{S}_i \subseteq \mathrm{app}(s_0, \mathcal{E}, \pi^e_0, \ldots, \pi^e_i) && \text{and} \\ &\mathrm{def}(\mathrm{app}(s_0, \mathcal{E}, \pi^e_0, \ldots, \pi^e_j)) \subseteq s_{j+1} \setminus \mathcal{S}_j && 0 < j \leq n \end{aligned}$$

$\mathcal{E}$ is an explanation for $\mathcal{S}$ iff it is an explanation for each $\mathcal{S}_i \in \mathcal{S}$.

Or less formally: an explanation are assumptions that, if they were true, would cause the observed behaviour of a surprise $\mathcal{S}_i$ (first condition), without explicitly contradicting other observations (second condition).

3.1 Explanation by omission

As it stands, there is a problem with this definition of explanation when trying to explain the *absence* of an effect. If an action has a conditional effect of the form $x = v \triangleright y = w$, explaining the absence of the effect on y may often be explained by *not* making an assumption on x. We want to present two possible ways of dealing with that problem, both involve additional forcing possible explanations $\mathcal{E}$ to contain a minimal set of assumptions:

- If $a \in \cup_{i=0}^{n} \pi^e_i$ is an executed action and $X \triangleright y = w$ is a conditional effect of a, then for each condition $x = v \in X$ there must be an assumption $e \in \mathcal{E}$ and a value $v' \in \mathcal{D}^x$ with $x = v' \in \mathrm{eff}(e)$.
- If $a \in \cup_{i=0}^{n} \pi^a_i$ is an assumption in the original plan with $x = v \in \mathrm{eff}(a)$, then there must be an assumption $e \in \mathcal{E}$ and a value $v' \in \mathcal{D}^x$ with $x = v' \in \mathrm{eff}(e)$.

The first approach forces there to be an assumption for every possible conditional effect that could have be triggered by an executed action. This will solve the "explanation by omission" problem, but requires that there *is* actually an assumption for every such condition in the planning domain. The second method is a bit more restricted, in that it only requires us to make or modify assumptions made during the original task. This may be sufficient, though, to explain the absence of conditional effects that were part of the original plan, and it is likely that assumptions for the involved variables exist (as they were part of the original plan).

3.2 Complexity

Theorem 1 *Let $\mathcal{S}$ be a set of surprises for a continual planning process $\mathcal{P}_\Pi$. Determining whether there is a set $\mathcal{E}$ that explain $\mathcal{S}$ is* NP*-complete.*

Proof.

Membership in NP: We first note that the number of assumptions per plan – and thus the size of $\mathcal{E}$ – is limited by the number of variables, $|\mathcal{V}|$, as each effect $x = v$ must be accompanied by a precondition $x = \perp$, and no other assumption can have $x = \perp$ as an effect. Verifying whether a given solution $\mathcal{E}$ is an explanation for $\mathcal{S}$ takes at most polynomial time, as checking applicability and computing action effects for each action is polynomial in $|\mathcal{V}|$ and the number of actions is linear in $|\mathcal{V}| + \sum_i |\pi_i^e|$.

Completeness: We show completeness by reduction from SAT. Given a set of variables $\theta_0, \dots, \theta_n$ and clauses $C_0, \dots, C_m$, we construct an explanation problem as follows:

- $\mathcal{V}$ contains one variable for each θ_i and for each clause C_j in addition to two variables G and F. $\mathcal{D}^v = \{\text{true}, \text{false}, \perp\}$ and $v = \perp \in s_0 \quad \forall v \in \mathcal{V}$.
- $\mathcal{A}^a$ contains two assumption a_i^t, a_i^f for each variable θ_i with $\text{eff}(a_i^t) = \{\theta_i = \text{true}\}$ and $\text{eff}(a_i^f) = \{\theta_i = \text{false}\}$. Additionally, if θ_i occurs positively/negatively in clause C_j, we add an assumption a_{ij}^C with $\text{pre}(a_{ij}^C) = \{\theta_i = \text{true/false}\}$ and $\text{eff}(a_{ij}^C) = \{C_j = \text{true}\}$. Finally, there is one assumption a^F with $\text{pre}(a^F) = \{C_j = \text{true}, 0 \leq j \leq m\}$ and $\text{eff}(a^F) = \{F = \text{true}\}$.
- The goal g is set to $G = \text{true}$
- $\pi_0^e = [a_0^e]$ contains exactly one action with no preconditions and a conditional effect $F = \perp \triangleright G = \text{true}$.
- The observed state after executing a_0^e, s_1, contains $G = \perp$.

An explanation for the surprise $\mathcal{S}_0 = \{G = \perp\}$ can only exist if there is a set of assumptions that make F true, which can only be done by finding a set of assumptions that correspond to a valuation of the variables θ_i that make all clauses true. ■

3.3 Adding diagnostic information

For performance or maintenance reasons it may be desirable to keep the complexity of the planning domain as low as possible. For this reason, it may be reasonable to use a separate, more detailed domain model for finding explanations. One of the most distinct features of a diagnosis domain is the use of *conditional effect*. Remembering the MOVE action from the introduction, it has a condition `(= (connected ?from ?to) true)`. For diagnostic purposes, though, the concept of hard preconditions is inconvenient, as they prevent the execution of the action completely. It is therefore necessary to replace those preconditions that are possible sources for failures with conditional effects:

```
(:action move
         :parameters (?r - robot ?from ?to - place)
         :precondition (= (is-in ?r) ?from)
         :effect (when (= (connected ?from ?to) true)
                       (assign (is-in ?a) ?to)))
```

The following formulation will ensure that, while the action can be executed as long as the robot is at the `?from` location, it will only have the desired effect if the two places are connected.

4 Finding Explanations as a Planning Problem

From the definition of surprises and explanations, it is quite obvious that a planner can be used for finding explanations. We create a new planning problem $\hat{\Pi} = \langle \hat{\mathcal{V}}, \hat{\mathcal{A}}, \hat{s}_0, \hat{g} \rangle$ that forces the planner to simulate the execution of the experienced task and requiring that each action's effect matches the observation.

Definition 7 (Observation action) *For a pair of observed and expected states, s_{i+1} and s'_i, and the identified surprises $\mathcal{S}_i$, the observation action o_i has the preconditions* $\text{pre}(o_i) = \mathcal{S} \cup \text{def}(s_{i+1})$ *and effects* $\text{eff}(\o_i) = s_{i+1} \setminus \text{pre}(o_i)$.

These observation actions o_i are placed between the executed plans π_i^e and π_{i+1}^e and enforce that the simulated execution of the plans results in the observed behaviour. Altogether, the new planning task contains the physical actions $\widehat{\mathcal{A}^p} = \{o_0, \ldots, \o_n\} \cup \pi_0^e \cup \ldots \cup \pi_n^e$. To force the planner to execute the actions in order, we use a helper variable v^e with $\mathcal{D}^(v_a) = \widehat{\mathcal{A}^p} \cup \{G\}$. We also set $v^e = a_0^e[0]$ in the initial state $\hat{s}_0$ with $a_i^e[j]$ denoting the j-th action of the i-th executed plan. We define a successor function succ on the actions as follows:

$$\text{succ}(a_i^e[j]) = \begin{cases} o_i & \text{if } j+1 = |\pi_i^e| \\ a_i[j+1] & \text{else} \end{cases}$$

$$\text{succ}(o_i) = \begin{cases} G & \text{if } i = n \\ a_{i+1}[0] & \text{else} \end{cases}$$

We then set the goal $\hat{g}$ of the new task to $v^e = G$ and augment every action $a \in \widehat{\mathcal{A}^p}$ as follows:

$$\begin{aligned} \text{pre}(a) &\leftarrow \text{pre}(a) \cup \{v^e = a\} \\ \text{eff}(a) &\leftarrow \text{eff}(a) \cup \{v^e = \text{succ}(a)\} \end{aligned}$$

As we retain the restrictions on assumptions that they may only be executed before any physical action, the planning problem only has a solution if the planner can find a sequence of assumptions that allow all originally executed actions to be executed in sequence, and that will result in the observations $\mathcal{S}_i$ that we identified as surprises.

If we want to enforce the planner to make assumptions for certain variables $x \in \mathcal{V}^{\mathcal{E}}$ to prevent explanations by omission as described in section 3.1, we can add for each such x a new variable x', with $\mathcal{D}^{x'} = \{\top, \bot\}$ and a set of assumptions that make x' true if some assumption on x was made: $a^v_{x'}, \text{pre}(a^v_{x'}) = \{x = v\}, \text{eff}(a^v_{x'}) = \{x' = \top\} \quad \forall v \in \mathcal{D}^x, v \neq \bot$ and add $x' = \top$ to the goal.

5 Object search example

We illustrate the process with an object search example from the Dora robot [Hanheide *et al.*, 2011]. The purpose of the robot is to find an object, for example a cereal box, and has probabilistic background knowledge about occurrences of objects in certain types of rooms. In this instance, this background knowledge may provide the information that cereals are frequently found in kitchens. This knowledge, together with probabilistic information about the type of the known rooms, is presented as assumptions to the planner, which then selects a probable set of assumptions that make it possible to reach the goal. If Dora starts out in `room0`, a minimal plan π_0, including those assumptions, could look as follows:

```
(assume-kitchen room1)
(assume-object-in-room cereal-box room1 kitchen)
(move dora dora room1)
(look-for-object dora room1 cereal-box object0)
```

The first two actions are assumptions that establish a probable state: That `room1` is a kitchen and that a cereal box is in that room. The `look-for-object` action will search the room for an object, `object0` of that type, providing the robot with the exact location if the object is actually there:

```
(:action search-for-object
         :parameters (?a - robot ?r - room ?l - label
                      ?o - object)
         :precondition (and (= (label ?o) ?l)
                            (= (is-in ?a) ?r))
         :effect (when (= (object-exists ?l ?r) true)
                       (assign (known-location ?o) true)))
```

If, after executing the look action, `(known-location object0)` is not true, the plan is no longer valid: The rest of the plan, π_0^u, is empty and the current state does not satisfy the plan. Let us say that the robot tries to look for the object twice more, before giving up completely. The complete planning process would look as follows:

$$\begin{aligned}
\pi_0 &= \big[\texttt{(A-kitchen room1), (A-obj-in-room obj0 cereal-box room1 kitchen),}\\
&\qquad\texttt{(move dora room1), (look-for-object dora room1 object0)}\big]\\
\pi_1 = \pi_2 &= \big[\texttt{(A-kitchen room1), (A-obj-in-room obj0 cereal-box room1 kitchen),}\\
&\qquad\texttt{(look-for-object dora room1 object0)}\big]\\
\pi_0^a = \pi_1^a = \pi_2^a &= \big[\texttt{(A-kitchen room1), (A-obj-in-room obj0 cereal-box room1 kitchen)}\big]\\
\pi_0^e &= \big[\texttt{(move dora room1), (look-for-object dora room1 object0)}\big]\\
\pi_1^e = \pi_2^e &= \big[\texttt{(look-for-object dora room1 object0)}\big]\\
s_0 &= \big\{\texttt{(= (is-in dora) room0)}\big\}\\
s_1 = s_2 = s_3 &= \big\{\texttt{(= (is-in dora) room1)}\big\}
\end{aligned}$$

The set of failures after all three plan executions is the same:

$$\mathcal{S}_0 = \mathcal{S}_1 = \mathcal{S}_2 = \big\{\texttt{(= (known-location object0) false)}\big\}$$

Any explanation for these failures must prevent the conditional effect of the `search-for-object` action from triggering. If we prevent explanations by omission, we must assume that `(object-exists cereal-box room1)` is false. There are several ways this could be achieved by the planner: If the probability of finding cereals in a kitchen is low to begin with, the explanation may simply be that cereals are not this kitchen. If the probability of `room0` being a kitchen is low, the explanation might be that the room is a living room and that cereals are usually not found there.

By adding more diagnostic information to the domain, a more varied set of explanations is possible. If we want to add the information, that small objects can be inside other objects and thus be hidden from direct observation, we could do so my changing the search action and adding some possible assumptions:

```
(:action search-for-object
         :parameters (?a - robot ?r - room ?l - label
                      ?o - object)
         :precondition (and (= (label ?o) ?l)
                            (= (is-in ?a) ?r))
         :effect (when (and (= (object-exists ?l ?r) true)
                            (= (directly-visible ?o) true))
                       (assign (known-location ?o) true)))
```

```
(:action assume-object-in-object
         :parameters (?l ?l2 - label ?o - object ?r - room)
         :precondition (and (= (label ?o) ?l2)
                             (= (size ?l) large)
                             (= (size ?l2) small)
                             (= (object-exists ?l ?r)
                             (= (object-exists ?l2 ?r)))
         :effect (assign (directly-visible ?o) false))
```

The `assume-object-in-object` allows us to assume that an object is not directly visible if it is small and a large object of some type exists in the room (note that we also need an opposite assumption, which is not shown here). The search action's effect is extended with the condition that the object must be visible.

With the added possibilities the following explanation could be given if the likelihood of finding cereals in `room1` is high enough:

$$\mathcal{E} = \big[\texttt{(A-kitchen room1), (A-object-in-room object0 cereal-box room1 kitchen),}$$
$$\texttt{(A-object-in-room object1 basket room1 kitchen),}$$
$$\texttt{(A-object-in-object basket cereal-box object0 room1))}\big]$$

6 Related Work

We have previously performed work that uses the approach of making modification to the initial state of a planning problem to find reasons for failing to come up with a plan [Göbelbecker *et al.*, 2010]. Sohrabi *et al.* [2011] use a similar approach to find explanations for a given sequence of observations, but do not integrate the process of finding explanations with a continual planning approach as we do.

7 Conclusion

We have formalised the notion of surprises and explanations in a continual planning problem, and shown that finding those explanations can be effectively translated into a new planning problem. using this approach, we can re-use the *assumption* formalism we use for the original planning process and apply it directly to the task of explaining surprises.

References

[Bäckström and Nebel, 1995] C. Bäckström and B. Nebel. Complexity results for SAS$^+$ planning. *Comp. Intell.*, 11(4):625–655, 1995.

[Göbelbecker *et al.*, 2010] Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI Press, may 2010.

[Hanheide *et al.*, 2011] Marc Hanheide, Nick Hawes, Charles Gretton, Hendrik Zender, Andrzej Pronobis, Jeremy Wyatt, Moritz Göbelbecker, and Alper Aydemir. Exploiting Probabilistic Knowledge under Uncertain Sensing for Efficient Robot Behaviour. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2011. to appear.

[Sohrabi *et al.*, 2011] Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. Preferred explanations: Theory and generation via planning. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI-11)*, pages 261–267, San Francisco, USA, August 2011.