Artificial Intelligence 247 (2017) 119–150

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Robot task planning and explanation in open and uncertain worlds

Marc Hanheide [*,1], Moritz Göbelbecker, Graham S. Horn, Andrzej Pronobis, Kristoffer Sjöö, Alper Aydemir, Patric Jensfelt, Charles Gretton, Richard Dearden, Miroslav Janicek, Hendrik Zender, Geert-Jan Kruijff, Nick Hawes, Jeremy L. Wyatt

ARTICLE INFO

Article history:
Received in revised form 14 August 2015
Accepted 18 August 2015
Available online 22 August 2015

Keywords:
Open-world planning
Planning under uncertainty
Commonsense knowledge
Mobile robotics
Probabilistic reasoning
Failure explanation
Assumptive planning
Retaskability

ABSTRACT

A long-standing goal of AI is to enable robots to plan in the face of uncertain and incomplete information, and to handle task failure intelligently. This paper shows how to achieve this. There are two central ideas. The first idea is to organize the robot's knowledge into three layers: instance knowledge at the bottom, commonsense knowledge above that, and diagnostic knowledge on top. Knowledge in a layer above can be used to modify knowledge in the layer(s) below. The second idea is that the robot should represent not just how its actions change the world, but also what it knows or believes. There are two types of knowledge effects the robot's actions can have: epistemic effects (I believe X because I saw it) and assumptions (I'll assume X to be true). By combining the knowledge layers with the models of knowledge effects, we can simultaneously solve several problems in robotics: (i) task planning and execution under uncertainty; (ii) task planning and execution in open worlds; (iii) explaining task failure; (iv) verifying those explanations. The paper describes how the ideas are implemented in a three-layer architecture on a mobile robot platform. The robot implementation was evaluated in five different experiments on object search, mapping, and room categorization.

1. Introduction

A long-standing challenge robotics poses for AI is how to act in uncertain and unfamiliar environments. As an example, imagine a robot that is switched on in an unfamiliar building, without a map, and given the task of finding a particular object. Questions that roboticists must answer include the following: *How should the robot integrate different kinds of information, such as commonsense knowledge and sensory input? How can the robot plan in the face of uncertain knowledge? How can the robot plan to achieve the task when it does not know about all the required objects and places? How can the robot deal with task failure intelligently?* This paper gives answers to all these problems, using two core ideas. The first idea is to organize knowledge into three levels, such that knowledge in a higher level enables reasoning about knowledge at lower levels. The second idea is to give the robot models of the effects of its actions on what it knows. One class of knowledge-modifying actions is assumptive actions. We show how planning with assumptions, combined with layered knowledge, solves several problems

[*] Corresponding author.
E-mail address: marc@hanheide.net (M. Hanheide).
[1] Authors Hanheide, Göbelbecker and Wyatt are identified as joint first authors.

http://dx.doi.org/10.1016/j.artint.2015.08.008
0004-3702/© 2015 Elsevier B.V. All rights reserved.

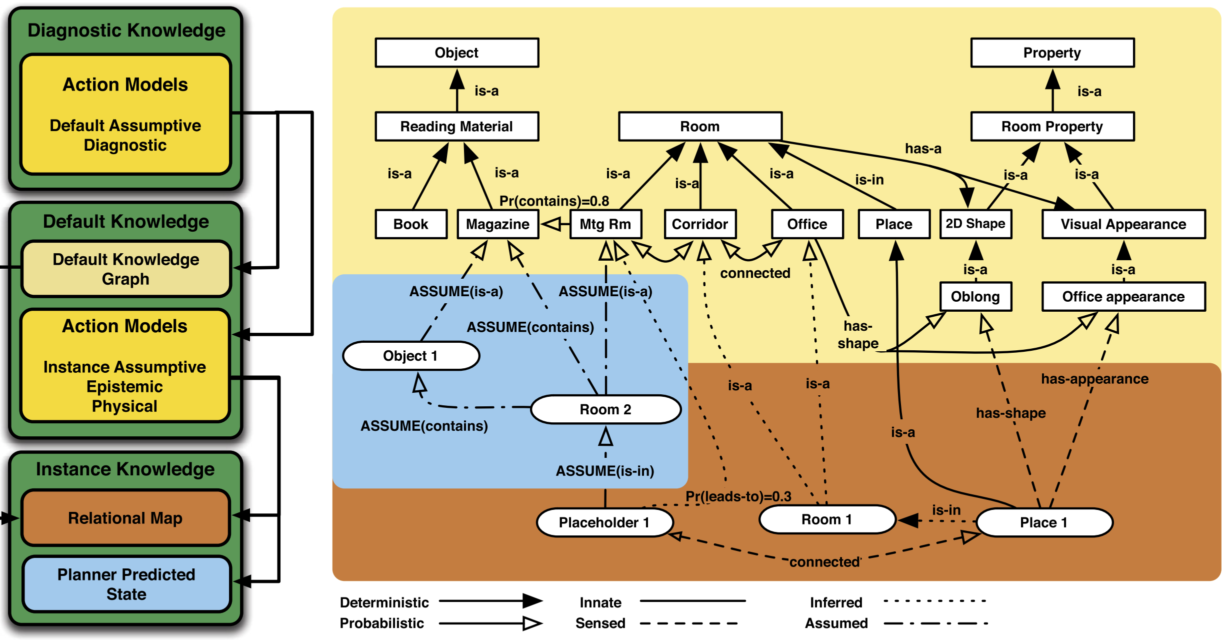


Fig. 1. (Left) The instance-default-diagnostic (IDD) knowledge schema, showing action models (yellow), general knowledge (cream), instance knowledge about the current state (brown), and predicted state (blue). **(Right)** Example state information in the concrete instantiation of the IDD schema used in this paper. The blue box represents possible instance knowledge assumed to be true by the planner in order to achieve the robot's goal (find a magazine). The right panel summarizes the robot's belief state at the start of the running example (see Fig. 3). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

in AI for robotics: (i) planning and acting under uncertainty, (ii) planning and acting in open worlds, (iii) explaining task failure, and (iv) verifying explanations.

1.1. IDD—a schema for robot knowledge

We propose a three-layer organization of knowledge: *instance*, *default*, and *diagnostic* (Fig. 1). We refer to this idea using the term *IDD schema*. Knowledge at a higher level is used to modify knowledge at lower levels. Knowledge is of two types: representations of state, and representations of the effects of actions on state. The *instance layer* contains only state information, describing the current environment. This state information includes the locations and categories of specific objects, rooms, and places in the current building, the whereabouts of particular people and what they know. The *default layer* contains general knowledge about categories—for example, which types of objects are typically found in kitchens. The default layer also contains knowledge of action effects on the instance state. This could be knowledge of the physical effects of actions, such as moving the robot, or of the knowledge effects of actions, such as asking where a particular object is. These knowledge-modifying (or epistemic) actions also include the ability to make *assumptions* about the existence of specific object instances—for example, assuming that there is a dining room in a particular house. Finally, the *diagnostic layer* contains only action knowledge. First, it contains assumptive actions for creating new general knowledge—hypothesizing that cornflakes boxes can often be found in dining rooms. Second, it contains the action models from the default layer augmented with possible causes of failure—the robot could not see the cornflakes box because it was hidden inside something. These two types of diagnostic knowledge can be used in conjunction with the other knowledge layers to help explain task failure. The robot could, for example, explain that it cannot find the cornflakes box because someone put it in a cupboard that it assumes is in the dining room. This diagnostic knowledge can also be used to hypothesize new default knowledge: the robot could hypothesize that cornflakes boxes are often kept in cupboards in dining rooms. Default and diagnostic knowledge are separated to enable efficient reasoning: simpler default knowledge is used most of the time, and more complex diagnostic knowledge is used only when needed.

1.2. An instantiation of the IDD schema

The schema above could be implemented in many ways, depending on the representations chosen. In this paper, the following choices have been made (Fig. 1) for the implementation of our robot system called *"Dora"*. First, in the in-

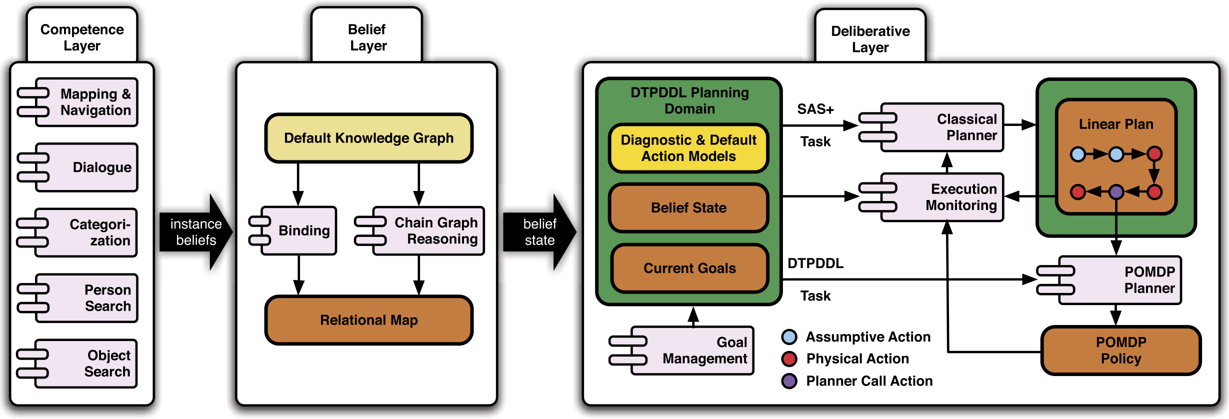


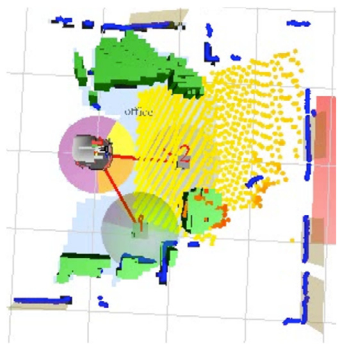
Fig. 2. The overall architecture shows how the belief-state maintenance and the switching planner are structured and linked to the basic competences for sensing and action. The implementation of this architecture is described in Section 6.

stance layer the current state is what we call a *relational map* (brown), represented as a libelled graph.[2] The nodes correspond to physical entities (objects, places) present in the current environment. The arcs represent relations between, and properties of, these entities. These arcs can be deterministic or probabilistic.[3] For example, the probability that *Placeholder1* leads to a meeting room is inferred to be 0.3. The instance layer also contains possible facts that the robot assumes to be true (blue). Each assumption inherits the probability of its directly supporting fact(s), so the assumption *ASSUME(leads-to(placeholder1,meetingroom))* also has probability 0.3. General knowledge about state is captured in the *default-knowledge graph* (cream) with use of the same formalism as for the relational map. In Fig. 1, a meeting room is a subtype of room for example, and by default the probability[4] that a meeting room contains a magazine is 0.8. Since the relational map and the default knowledge are both relational graphs, they can be linked to create a single graph. This makes joint inference over the two layers possible with use of a single inference mechanism. Critically, it also makes it possible to imagine unseen entities in the relational map by use of the default-knowledge graph as a generative model for elements in the relational map. For example, if you think a room is likely to be a kitchen, you can assume the existence of instances of common kitchen objects in that room. In Fig. 1 the predicted instance state (blue) consists entirely of assumptive knowledge, but in general predicted instance states will contain a mix of assumptions (Dora assumes the magazine is in the office), physical information (the magazine is in the office), and epistemic information (Dora knows that the magazine is in the office). Using knowledge in the diagnostic layer, one can also make assumptions predicting new default knowledge. Further details of the specific state representations employed in the instance and default layers will be given in Section 3, and further details of the action representations will be given in Section 4.

1.3. A three-level architecture

In addition to representations, an architecture is required (Fig. 2). A central problem in robotic architectures is how to design a system to acquire a stable representation of the world suitable for planning and reasoning. Events occur rapidly and concurrently, in each of the robot's sensory and action modalities. Our approach is to employ a distributed blackboard system to control the flow of information. This distributed system is organized into three architectural layers that are independent of the three knowledge layers described above. At the lowest architectural layer, called the (*competence layer*), a group of processes related to a specific modality or skill (e.g. dialogue, or mapping and navigation) share information with each other via a blackboard or working memory. They transmit abstracted summaries of that information, designed to be temporally stable, to the *belief layer*. This middle layer has processes that link, or bind, information from different competences and modalities. This binding creates a set of symbols, describing the instance state—here the relational map—that can support task planning in the *deliberative layer* of the architecture.[5] Using this method, we have designed many retaskable robot systems, which reuse and share many components. The Dora system described in this paper comprises more than 100 concurrently running components. We refer the reader to more detailed descriptions of our architectural approach elsewhere, as while important, it is not the focus of this paper [16,19,17].

[2] Or equivalently a set of ground atomic formula in first-order logic.
[3] Clearly deterministic relations are simply probabilistic relations with probability 1.
[4] How these probabilities are acquired is beyond the scope of this paper; see Section 6.4 for a brief overview.
[5] In previous work [6], we have referred to these as amodal symbols. We use the term amodal to reflect the fact that they are independent of any specific modality.



Action	Cost	Probability	Status
leads-to-room-placeholder1-meetingroom	0	0.30	Done
room-from-placeholder placeholder1 room2 meetingroom	0	1.00	Done
object-in-room magazine room2 meetingroom	0	0.80	Done
virtual-object-position object1 magazine room2	0	1.00	Done
move dora place1 placeholder1	2	1.00	Running
create-cones dora magazine room2 placeholder1	5	1.00	
search-for-object dora magazine room2 placeholder1 object1	200	1.00	
Total:	**207**	**0.243**	

Fig. 3. **(Left)** This shows the initial map state when Dora is switched on. Details are given in the text. **(Right)** This shows the plan generated for the goal "find a magazine" from the initial state. Light blue entries are assumptive actions, green denotes the current action, white denotes an unexecuted action, and orange denotes a planned call to the decision-theoretic planner. The probabilities of success for each step are taken, by the planner, from the relational map. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

1.4. Road map

Having sketched the idea of knowledge layers, and their instantiation, we show in Section 2 how they are used with planning, by means of an example that will be used throughout the paper. Following this, Section 3 describes the state representation (relational map, default knowledge graph) in detail. This state representation is able to integrate knowledge from different sources such as visual sensing, dialogue, and common sense. Sections 4 and 5 then describe the action representation, and the planning approach able to reason with the action and state representations to solve problems (i)–(iv). An overview of the robot implementation is given in Section 6, and Section 7 gives results on the performance of the solutions to problems (i)–(iv).

2. An illustrative example

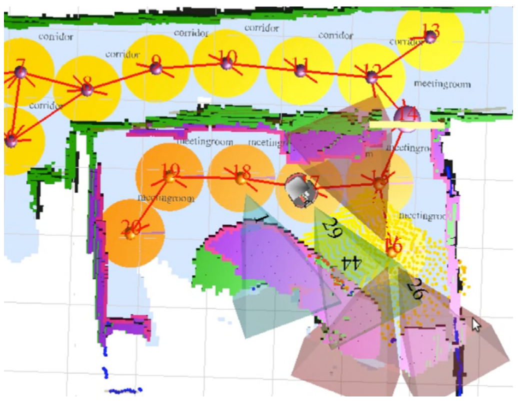
We start with an example run of the robot. Dora is switched on in an unfamiliar office building and given the task[6] of finding an object: a copy of *AI Magazine*. Dora has the following sensing abilities: it can detect objects and people using vision; it can map free space with a laser and a depth camera; it can acquire information about objects and rooms via dialogue with a human; and it can sense the 2D shape and visual appearance of a room. The initial map state is shown in Fig. 3 (left). In this, blue dots are laser readings, light blue marks free space, yellow-red dots indicate obstacle height, green areas are raised horizontal surfaces and grey planes are wall segments fitted to the laser data. When switched on, the robot thus obtains a local occupancy map, and the first node (*place1*) in the relational map is created at its current location, as shown in Fig. 3 (left), and in Fig. 1 (brown panel). This is a *place* node, and two empty *place-holder* nodes are created in yet to be visited locations. Each place-holder is predicted to turn into a full place node when visited. Each place node has an associated visual appearance, and a 2D shape for the surroundings. These appearance and shape categories are obtained, respectively, from visual and laser-based categorizers [35].

The room category is then determined via Bayesian inference on the relational map, guided by the default-knowledge graph. The latter represents the hierarchy of room types (meeting rooms, offices etc.), and object types (magazines, tables, containers, etc.). It encodes commonsense room type-object type relations, expressed probabilistically ($\Pr(\mathit{contains}(\mathit{meetingroom}, \mathit{magazine})) = 0.8$); probabilities of visual and 2D appearance for room types; and the probabilities of direct connectivity between room types ($\Pr(\mathit{connected}(\mathit{office}, \mathit{corridor})) = 0.95$). The object-room relations enable a room's category to be inferred from object detections ("I see a fridge and a microwave so this is a kitchen with probability 0.6."). In the example, visual appearance and shape combine with default knowledge to give the distribution over the category of *room1*: $\langle \Pr(\mathit{office}) = 0.52,\ \Pr(\mathit{corridor}) = 0.24,\ \Pr(\mathit{meetingroom}) = 0.24 \rangle$. In Fig. 3, the pie chart attached to the node for *place1* shows this distribution, where purple is office, yellow is corridor, and orange is meeting room. The resulting relational map constitutes the robot's *belief state*, used as the initial belief state for planning. After obtaining the initial relational map, Dora is given a goal description that *a magazine must exist and the robot must know where it is*, written:[7]

```
(exists (?o - object) (and (= (label ?o) magazine) (K (position ?o))))
```

[6] We use the terms task and goal interchangeably throughout the paper.

[7] Throughout the paper, we use `typewriter font` to denote excerpts from our planning domain, encoded some variant of a planning domain description language (PDDL). This includes goals and action operators.



Action	Cost	Probability	State
object-in-room magazine room2 meetingroom	0	0.80	Done
virtual-object-position object1 magazine room2	0	1.00	Done
move-direct dora place2 place3	3	1.00	Done
move-direct dora place3 place5	3	1.00	Done
move-direct dora place5 place8	3	1.00	Done
move-direct dora place8 placeA	3	1.00	Done
move-direct dora placeA placeC	3	1.00	Done
move-direct dora placeF placeF	3	1.00	Done
create-cones dora magazine room2 placeF	5	1.00	Done
search-for-object dora magazine room2 placeF object1	200	1.00	Running
Total:	**223**	**0.8**	

Fig. 4. (Left) The map state, three minutes into a run, illustrates the view-planning problem handled by the POMDP planner. The purple tetrahedra are the view cones. **(Right)** The corresponding plan execution status; grey actions represent completed actions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2.1. Planning with hypotheses

To create a plan to find the magazine, Dora must reason about two entities that do not yet exist in its relational map: the magazine and the room that object is in. This incompleteness in knowledge requires open-world planning—that is, planning that takes into account the fact that new entities may become known to the robot during plan execution. Given the initial relational map, the planner hypothesizes extensions to it. To create these extensions, the planner has *assumptive actions* it can take, each making one or more assumptions about instance knowledge. The planner selects a sequence of assumptive actions that are likely to be true according to default knowledge, and which enable goal achievement. The planner does not make task-irrelevant assumptions. In Fig. 1 (blue panel), five assumptions have been made in this way: (i) a meeting room exists; (ii) there is a route from a place-holder to the hypothesized meeting room; (iii) this meeting room contains objects of type magazine; (iv) a particular instance of type magazine exists; and (v) it exists in this instance of a meeting room. The remainder of the plan (Fig. 3) is to move to the hypothesized meeting room and to plan to conduct a visual search for the object when the robot gets there. The task-driven selection of assumptions is the key step in dealing efficiently with incomplete information, and we describe our approach to it in detail in Section 4.

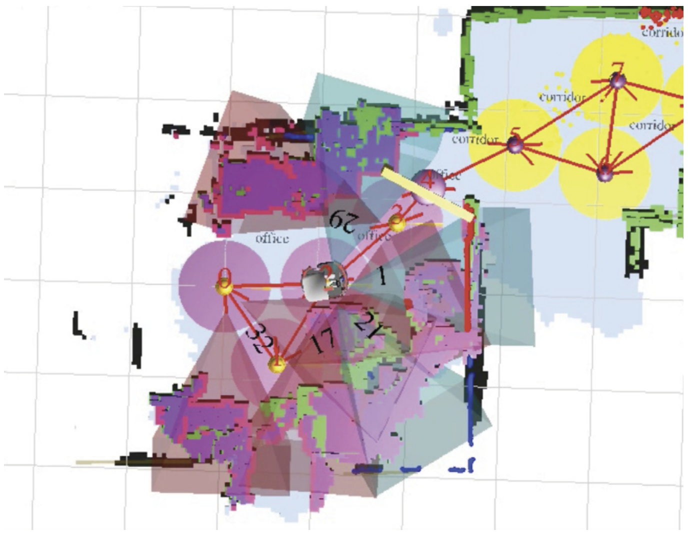
Next, the first physical plan step, `(move place1 placeholder1)`, is executed. When the robot reaches the place-holder, the relational map changes. This invalidates the current plan—because, for example, the place-holder does not belong to a new room—and so execution monitoring triggers replanning. This cycle of planning, execution, and belief updating continues until either the goal is achieved or a plan cannot be found (see the rightmost box in Fig. 2).

2.2. Planning under observational uncertainty

The continual planner is actually composed of two planners: a classical planner (Fast Downward [18]), and a decision-theoretic (partially observable Markov decision process, POMDP) planner. The classical planner also makes decisions about when to make a call to the POMDP planner, and thus we refer to the whole planning system as a switching planner. The POMDP planner is advantageous for reasoning about situations with uncertain belief state and actively controlled but unreliable perception. The continual planner uses assumptive actions to create the POMDP for the POMDP planner to solve, as described in Section 4. In Fig. 4 (three minutes into the run) Dora has found a meeting room, created a set of visual fixations, and is executing a POMDP plan that sequences these. If the robot sees the magazine with sufficient confidence, it will terminate in success. If it fails to find the magazine in the meeting room, Dora will replan and search for it in other rooms—in this case the office where the robot started. If it does not find the object, it will eventually declare a task failure. This happens when the relational map has a low probability that the magazine exists in any of the existing rooms, and there are no unexplored place-holders that could lead to other rooms. Because of this, the likelihood of the best sequence of assumptions—consistent with the object being findable—is sufficiently low that there is no plan with an acceptably low cost. Task failure is thus declared. Fig. 5 (left) shows the map at this point.

2.3. Explaining unexpected failure

Whenever a task failure is declared, Dora raises a new goal to explain it. The sequence of plans gave a sequence of expected observations (the robot should see the magazine in the meeting room, or failing that in the office). In the case of task failure, these observations did not occur, so there was a mismatch between expectation and experience. We refer to the difference between the two as the set of surprises. The aim is to explain the surprises that caused the task failure. This problem is closely related to explanation-based learning, where explanations are generated by planning with a domain theory [30]. We pose it as the problem of finding an additional set of assumptions that change



Action	Cost	Probability	State
contains-person-room2-true	0	0.20	Done
entity-exists-object7-true	0	1.00	Done
person-in-room person1 placeF room2	0	1.00	Done
move dora place2 place3	2	1.00	Done
move-direct dora place3 place5	3	1.00	Done
move-direct dora place5 place8	3	1.00	Done
move-direct dora place8 placeA	3	1.00	Done
move-direct dora placeA placeC	3	1.00	Done
move-direct dora placeC placeF	3	1.00	Done
look-for-people dora person1 placeF	10	1.00	Running
engage dora person1	1	1.00	
ask-for-object-existence dora person1 container object7 in room2	10	1.00	
ask-for-dk-inobject dora person1 magazine container	10	1.00	
ask-for-dk-inroom dora person1 container meetingroom	10	1.00	
ask-for-object-existence dora person1 magazine object6 in object7	10	1.00	
Total:	**68**	**0.2**	

Fig. 5. (Left) The robot in the office, at the time of failure. The view cones searched are coloured blue. The robot is in the centre of the room. **(Right)** The plan to verify the explanation of the failure, showing the three assumptions that need to be made. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the expected outcomes to make them consistent with the actual observations. *Diagnostic knowledge* is used to enable these new assumptions to be made. There are two types of diagnostic knowledge. The first type extends the action models from the default layer with knowledge of how those actions might fail. These failure modes are represented with use of what are termed *conditional effects* in the planning community. The resulting, more complex action models would make normal planning more time-consuming, and so are used only when unexpected failures occur. The second type of diagnostic knowledge is a new set of assumptive actions that can hypothesize new pieces of default knowledge. Using these two new kinds of actions, Dora tries to find a modified version of the plan that consists of the original plan, preceded by a new sequence of additional assumptions. Addition of these assumptions in front of the old plan results in the previously surprising outcomes becoming expected outcomes. In this run the explanation is as follows:

1. Meeting rooms have containers by default: `(new-dk-inroom container meetingroom)`.
2. Containers have magazines inside them by default: `(new-dk-inobject magazine container meetingroom)`.
3. There is a container in the meeting room: `(object-in-room-new container room2 meetingroom)`.
4. There is a magazine in the container: `(object-in-object-new magazine container object1 room2 meetingroom)`.

All of these actions are assumptions. Assumptions 1 and 2 hypothesize new default relations between objects and places (default assumptions), and assumptions 3 and 4 hypothesize new instance knowledge (instance assumptions). Having devised this explanation, Dora creates a goal to verify each assumption. A plan is created for the conjunction of these goals, as shown in Fig. 5 (right). Dora can also make assumptions in this new planning process. In this case, Dora hypothesizes the existence of a person, who can be asked questions in order to verify each part of the explanation. When the person is found, Dora asks questions to verify the assumptions:

Q1: Is there a container in this room? `(ask-for-object-existence dora container object7 in room2)`
Q2: Is there a magazine in the container? `(ask-for-object-existence dora magazine object6 in object7)`
Q3: Are magazines typically in containers? `(ask-for-dk-inobject dora magazine container)`
Q4: Are containers typically in meeting rooms? `(ask-for-dk-inroom dora container meetingroom)`

If Dora receives a positive answer to a question, it can update the relevant knowledge. In the case of Q1 and Q2, this means updating the relational map. For Q3 and Q4, it means updating the default knowledge. Thus, Dora can plan and verify explanations that lead to updating of both instance and default knowledge. This example has sketched how problems (i)–(iv) are solved. We now detail each aspect of the solution, starting with the state representation.

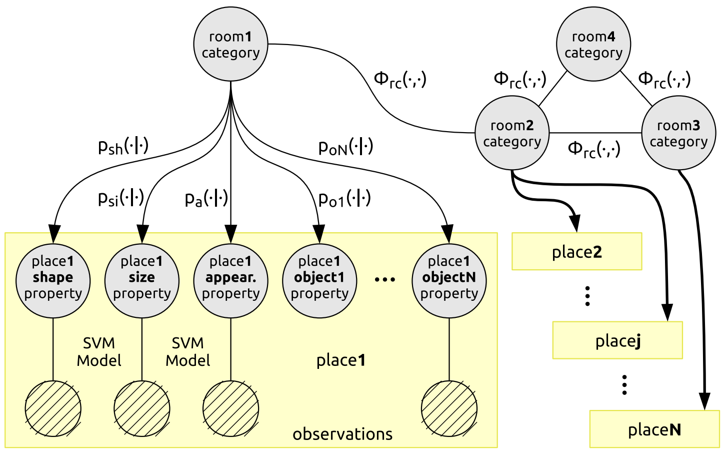


Fig. 6. This shows the structure of the chain graph model, compiled from the relational model. The vertices represent random variables. The edges represent the directed and undirected probabilistic relationships between the random variables. The textured vertices indicate observations that correspond to sensed evidence.

3. Representing instance and default state

Fig. 1 shows the relational map (brown) and the default-knowledge graph (cream) for the running example.[8] The relational map has already been described. In addition to the already mentioned features, the default-knowledge graph also includes "is-a" for type relations, and "has-a" relations for spatial properties such as shape, size or appearance, and for room-object associations.

3.1. Inference and state estimation

To allow (Bayesian) inference across the default-knowledge graph and the relational map, the joint representation is compiled into a *chain graph* [35], the structure of which is adapted on the fly according to the state of the underlying belief models. Chain graphs provide a natural generalization of directed (Bayesian network) and undirected (Markov random field) graphical models, allowing us to model both "directed" causal (such as "is-a" relations) and "undirected" symmetric or associative relations (such as connectivity of room types). The use of a chain graph allows us to model circular dependencies originating from loops in the topological graph, as well as permitting direct use of the probabilistic relations between the concepts. In our implementation, chain graph inference is event driven, triggered by updates to the relational map. For example, if an appearance property or object detection alters the probability of a relation, inference propagates the consequences throughout the graph. In our work, the underlying inference is approximate, and uses the fast loopy belief propagation procedure [31]. A chain graph supporting the running example of Figs. 1–5 is shown in Fig. 6. Each place node is represented by a set of random variables, one for each relation it is involved in. These are connected to a random variable over the room category, representing the "has-a" relations between rooms and their properties. The room category variables are connected by undirected links to one another, according to the actual connectivity of instances of rooms in the relational map. The potential functions $\phi_{rc}(\cdot,\cdot)$ come from the default knowledge about connectivity of room types. The remaining variables represent the shape and appearance properties of the surrounding space, as observed from each place node, and also the presence of objects. These are connected to observations of features extracted directly from the sensory input. As explained in Section 6.2.2, these links are quantified by the categorical models of sensory information. Finally, the distributions $\Pr_s(\cdot|\cdot)$, $\Pr_a(\cdot|\cdot)$, and $\Pr_{o_i}(\cdot|\cdot)$ represent the default knowledge about shape, appearance and object co-occurrence, respectively. They allow for inference about other properties and room categories—for example, that the room is likely to be a kitchen, because you probably observed cornflakes in it. After inference, the resulting combined graph provides the *belief state* for planning (Fig. 2). Thus, when we refer to belief state, we mean not only the instance state, but also the default knowledge. The processes that maintain this belief state integrate evidence, arriving asynchronously, from many concurrent sensing processes. Thus, in architectural terms, as opposed to knowledge terms, we also have three-layers. This three-layer architecture, shown in Fig. 2, will be described in more detail in Section 6.

4. Planning

There are two approaches to task planning we might take: classical planning and decision-theoretic planning. Classical planning is fast, but cannot reason about multiple action outcomes, unreliable observations, or multiple possible worlds.

[8] The relational map and default-knowledge graph were originally presented in [35,38], where they were termed the *conceptual map*.

It thus produces a linear plan, which succeeds only if all the actions in the plan succeed. Decision-theoretic planning, by contrast, produces a policy maximizing the probability of success[9] from any belief state the robot might reach during plan execution, taking account of the probabilities of every possible action outcome in those states. Decision-theoretic planning thus handles multiple action outcomes (Markov decision processes, MDPs) and unreliable observations (POMDPs), but at an unfeasible computational cost given the size of typical robot domains. We therefore developed a continual switching planner [11], which switches between a deterministic planner [18] and a POMDP solver, as required.[10] This creates a linear plan, with some actions in that plan being calls to a decision-theoretic (POMDP) planner. This enables us to solve large problems quickly, while utilizing decision-theoretic planning where its benefits are greatest.

As mentioned previously, there are two concepts necessary to enable deterministic planning in incomplete and uncertain worlds: *knowledge-level predicates* and *assumptive actions*. Two knowledge-level predicates [33] model the robot's epistemic state over time: a knowledge predicate and an assumptive predicate. Physical actions (such as sensing actions) can have knowledge predicates as effects, as described in Sections 4.2 and 4.3. Assumptive actions, discussed in Section 4.5, are by contrast "virtual actions" that only have assumptive predicates as effects, establishing possible, but uncertain facts about the world. An assumptive action succeeds with a probability equal to the probability of the facts it establishes according to the belief state. Assumptive actions always occur at the beginning of a plan, and are used to satisfy preconditions for physical actions later in the plan. Assumptions may, and often do, depend on each other, allowing us to model non-trivial dependencies.

This brings us to the cost function that each part of the switching planner seeks to optimize. The POMDP solver simply finds the optimal policy. The linear plan succeeds with the probability of the sequence of assumptions it makes, and so is suboptimal in an uncertain world. Its degree of suboptimality cannot strictly be known without full decision-theoretic planning being performed. A reasonable upper bound on suboptimality is the difference between the goal reward, and that reward weighted by the probability of the linear plan succeeding. Finally, we need to account for the costs of the plan. Since the plan starts with uncertain assumptions, we cannot assign failure probabilities to each step of the deterministic physical plan. An upper bound on the linear plan cost, but a lower bound on the decision-theoretic cost, is achieved by incurring the full costs of all the actions in the linear plan. Given reward R for the goal state, this results in our trying to minimize the following:

$$c(\pi) = \sum_{a_i \in \pi} \mathrm{c}(a_i) + \left(1 - \prod_{a_i \in \pi} \rho(a_i)\right) R$$

where π denotes the linear plan, a_i denotes its ith action, $\mathrm{c}(a_i)$ denotes the cost of that action, and $\rho(a_i)$ denotes its probability of succeeding.[11] This formulation penalizes the linear plan according to a regret term, which discourages the sequential planner from scheduling unlikely assumptions. Typically only the assumptive actions have a non-zero probability of failure, and so in most of our examples $\prod_{a_i \in \pi} \rho(a_i)$ is precisely the probability of a particular world in which the physical part of the linear plan succeeds.

Algorithm 1 and Fig. 2 give an overview of the cycle of planning and execution. First, the deterministic planner creates a linear plan that achieves the goal. As part of this, it makes a sequence of assumptions to select a possible world in which the subsequent physical actions will succeed. If the outcome of the next physical action is non-deterministic,[12] a POMDP is created for that portion of the planning problem. This POMDP includes *goal actions*, which allow the POMDP planner to achieve a terminal reward by committing to a particular action effect to be communicated back to the continual planner. The robot then executes the policy generated by the POMDP planner until a goal action is reached (lines 13 ff.), before continuing with the execution of the rest of the linear plan. After each action in the linear plan has been executed, the relational map is updated. Execution will continue (line 21) until the goal is achieved or the plan becomes invalid, at which point replanning occurs.

4.1. Planning framework and notation

In the rest of this section, we detail the machinery that enables our planning approach to work in open and uncertain worlds, before showing how to use it to explain failures in Section 5. Our planning framework is based on PDDL 3.1, an extension of PDDL 3.0 [10]. To support decision-theoretic planning, we have added the probabilistic effects and initial state descriptions from Probabilistic PDDL [53] and our own extensions for describing probabilistic observation models. We refer to the resulting language as DTPDDL [11] (for "decision-theoretic PDDL").

In our implementation, we encode the planning domain using DTPDDL. This is translated into a representation called SAS$^+$ for the deterministic planner [4], while the POMDP solver uses the full DTPDDL representation. In most of the follow-

[9] Or more generally, the expected value.

[10] We refer to the continual switching planner as either a "continual planner" or a "switching planner", depending on the aspect to be emphasized.

[11] We use ρ to denote the *success probability* of a single action in our deterministic planning model, in which the only possible outcomes are success and failure. In contrast, we use $\Pr(\cdot)$ for probability *distributions* that are provided by the conceptual map. In our PDDL examples, the former (ρ) is written as an `(assign (probability))` effect and the latter (Pr) as a `(probabilistic)` expression in either an action effect, or initial state description.

[12] Typically in our examples this is the case for sensing actions.

Algorithm 1 Continual switching planner.

```
1: Input: initial state $s$, goal $g$
2: Output: success or failure
3: loop
4:     if IsGoalReached($s$) then
5:         return success
6:     end if
7:     $\pi \leftarrow$ CreatePlan($s, g$)
8:     if $\pi = \emptyset$ then
9:         return failure
10:    end if
11:    while $\pi$ is valid in $s$ do
12:        $a, \pi \leftarrow \pi$
13:        if $a$ has nondeterministic effects then
14:            $s^{DT} \leftarrow$ CreateAbstractState($s, \pi$)
15:            $\Pi^{DT} \leftarrow$ CreatePOMDP($s^{DT}, \pi$)
16:            $a \leftarrow$ POMDPPlan($\Pi^{DT}, \emptyset$)
17:            while $a$ is not a goal action do
18:                $s, o \leftarrow$ Execute($a$)
19:                $a \leftarrow$ POMDPPlan($\Pi^{DT}, o$)
20:            end while
21:        else if $a$ is a physical action then
22:            $s, o \leftarrow$ Execute($a$)
23:        end if
24:    end while
25: end loop
```

ing examples, we show action examples as lifted DTPDDL operators. We will also find it useful to explain some ideas using SAS$^+$ notation. Informally, SAS$^+$ is a planning formalism in which states are represented as a set of variables with finite domains, and action preconditions and effects consist of *assignments* of values to variables.[13] Formally, an SAS$^+$ task Π is a tuple $\langle \mathcal{V}, \mathcal{A}, s_0, g \rangle$ of variables, actions, an initial state, and a goal description. Each variable $v \in \mathcal{V}$ has a finite *domain* $\mathrm{dom}(v)$ with the *unknown* value $\perp$ contained in every domain. A state s can be regarded as a set of facts $v = x$ which assigns to each variable an element of its domain. By default, every variable is set to the unknown value, and we use $\mathrm{def}(s) = \{v = x \in s : x \neq \perp\}$ to denote the set of *defined* assignments of a state. An action $a \in \mathcal{A}$ has a precondition $\mathrm{pre}(a)$, effects $\mathrm{eff}(a)$, associated costs $\mathrm{c}(a)$, and probabilities $\rho(a)$. Preconditions are sets of assignments, and effects are sets of either simple assignments or conditional effects.

For reasons of brevity, we refrain from giving a formal definition for action and plan application, as we use the normal SAS$^+$ semantics. We refer to the result of applying action a in state s as $\mathrm{app}(s, a)$ and that of applying a sequence of actions $a_0, \ldots, a_n$ as $\mathrm{app}(s, a_0, \ldots, a_n)$. An example of an action operator is that for the move action:

```
(:action move
        :parameters (?a - robot ?from ?to - place)
        :precondition (and (connected ?from ?to)
                           (= (is-in ?a) ?from))
        :effect (and (increase (total-cost) 2)
                     (assign (probability) 1.0)
                     (assign (is-in ?a) ?to)
                     (assign (placestatus ?to) trueplace)
                     (K (in-room ?to))))
```

Its preconditions are that the robot has to be at the starting place `?from` and that `?from` and `?to` are connected. Executing the action causes the robot to be at `?to`, turns it into a place node, and has a knowledge effect which will be explained in the next section. There are two special effects in this action: `(increase (total-cost) 2)` assigns costs c of 2 to the action, and `(assign (probability) 1.0)` sets the action's success probability ρ to 1.0. If those effects are absent, action costs are 1.0 for physical actions and 0 for virtual actions (such as assumptions), and the default value for probabilities is 1.0. The use of `(assign (probability))` is semantically different from the use of probabilistic action effects such as in PPDDL (which are also used in the observation models described in Section 4.4). Instead of describing alternative action outcomes, action probabilities less than 1.0 increase the likelihood that the entire plan fails.

[13] In contrast to formalisms such as STRIPS, in which states are sets of logical atoms.

4.2. Representing knowledge state

We now detail the two *knowledge-level predicates* we use. The *knowledge* predicate $\mathbf{K}(v)$ indicates that the value of v is known.[14] The *assumptive* predicate $\mathbf{A}(v, x)$ states that the planner has made the assumption[15] that v has the value x. Knowledge predicates obey a few rules that are enforced by the planning system:

Definition 1. For a variable v and a value $x \in \text{dom}(v) \setminus \{\bot\}$, the following conditions hold in each state s:

- $(\exists x)\,(v = x) \in s \Rightarrow \mathbf{K}(v) \in s$.
- $(v = x) \in s \Rightarrow \mathbf{A}(v, x) \in s \wedge \left(\forall x' \neq x\right) \mathbf{A}(v, x') \notin s$.

For the initial state s_0 the first condition holds in both directions:

- $(\exists x)\,(v = x) \in s_0 \Leftrightarrow \mathbf{K}(v) \in s_0$.

The first condition means that if a variable is set to a value other than the unknown value, this variable must be known to the robot.[16] The second condition ensures assumptions cannot contradict existing variable assignments. The initial state s_0 is special in that knowledge cannot be asserted by setting $\mathbf{K}(v)$: if we knew the value of v, that fact has to be part of the initial state.

4.3. Representing the knowledge effects of actions

The knowledge predicate $\mathbf{K}(v)$ can be used to model the effects of sensing actions. For example, when the robot moves to a new place, it will know to which room the place belongs. This is modelled by the effect `(K (in-room ?to))` of the `move` operator. In this case, the actual value of the `in-room` variable is irrelevant for the outcome: we will gain the information in any case. For other sensors, the outcome may depend on the value of the variable we are trying to sense. For example, when searching for an object, even if sensing were deterministic, the robot would only learn the object location if it actually were where the robot looked. If we modelled sensing in our object search operator in the same way as in the `move` operator, the robot might end up looking in the wrong place repeatedly, because the knowledge effect asserts that executing the action will result in knowing the object's location unconditionally. This can be avoided by our making the knowledge effect a *conditional effect*. A conditional effect $C \triangleright e$ means that the effect e will occur only if C is true when the action is executed. In PDDL it is written with the `when` keyword. For our `search-for-object` action, this can be applied as follows:

```
(:action search-for-object
         :parameters (?a - robot ?l - label ?r - room
                      ?p - place ?o - visualobject)
         :precondition (and (= (is-in ?a) ?p)
                             (= (in-room ?p) ?r)
                             (= (label ?o) ?l)
                             (cones-created ?l in ?r))
         :effect (and (increase (total-cost) (search-cost ?l ?o))
                      (when (A (position ?o) ?r)
                             (K (position ?o)))))
```

The conditional effect forces the planner to look for the object in places where it can be assumed to be. Of course, for this to work, we require that the `(A (position ?o) ?r)` predicates have the correct values. A naive continual planner could simply make all assumptions that are sufficiently likely to be true. This would ensure that the robot searches only those rooms where the object is likely to be. Such a coarse approach, however, discards most of the probabilistic knowledge the system possesses. This is why we introduce *assumptive actions*, and allow them to inherit probabilities from the current belief state, as a way to discriminate between plans according to their probability of success.

Having described how knowledge and knowledge effects are represented, we need to use that knowledge during planning. In the simplest case, a knowledge-level predicate is an explicit precondition (as in the conditional effect above) or part of a goal such as the one given to Dora in our example:

```
(exists (?o - object) (and (= (label ?o) magazine) (K (position ?o))))
```

[14] This notion can be extended so that the planner can reason about the knowledge of multiple agents [5]. This is supported in our system.

[15] Our use of "assumption" differs somewhat from its everyday use in that we can assume a fact even if we know it to be true.

[16] In SAS$^+$ notation we write lack of knowledge as $\mathbf{K}(v) = \bot$. In DTPDDL this is equivalently written as $\neg$`(K v)`. We choose notation by context.

There are subtler uses of knowledge-level predicates. For example, the search action has the precondition that the robot must be in the room which it plans to search: `(= (is-in ?a) ?p)` and `(= (in-room ?p) ?r)`. But when planning in partially explored environments, we do not yet know to which room a place node belongs. Suppose, however, that we can make an assumption about a place-node's room, then the knowledge effect `(K (in-room ?to))` of the `move` action should allow us to use the search action in a plan. We achieve this by applying a simple transformation to all physical actions: whenever there is a precondition $(v = x)$, we replace it by the two knowledge-level conditions $\mathbf{K}(v)$ and $\mathbf{A}(v, x)$. This means that an action can be scheduled once suitable values for its preconditions are assumed and the preconditions are known. At the same time, plans that were valid before this transformation remain so: because a fact $(v = x)$ implies $\mathbf{K}(v)$ and $\mathbf{A}(v, x)$, the new action is applicable whenever the original action was. Also, the planner will never try to schedule an action whose preconditions are only satisfied by assumed knowledge. This is because in the initial state $\mathbf{K}(v)$ implies, by definition, that v has some known value assigned, and an assumption may not contradict that assigned value.

4.4. Probabilistic observation models

Knowledge-level predicates are sufficient to model the qualitative knowledge changes needed for continual planning. To allow decision-theoretic planning of sensing, DTPDDL adds *observation models* to PDDL. These describe the *signals* that the robot may receive when actions are executed, for example, for person search:

```
(:action look-for-people
         :parameters (?a - robot ?pl - place)
         :precondition (= (is-in ?a) ?pl)
         :effect (and (increase (total-cost) 10)))

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

The action `look-for-people` does not have any effects except to incur the action cost. However, the `person` observation model—acquired off line from data—specifies that whenever the `look-for-people` action is executed, the robot correctly detects each person present with probability 0.7, or generates a false positive with probability 0.001. These observation models can be used by the POMDP planner, but not the classical planner. We thus automatically translate them into epistemic effects as follows: if an assignment $v = x$ is part of an observation model's precondition or conditional effect, we add a conditional effect $\mathbf{A}(v, x) \triangleright \mathbf{K}(v)$ to the triggering action's effects:

```
(:action look-for-people
         :parameters (?a - robot ?p - person ?pl - place)
         :precondition (= (is-in ?a) ?pl)
         :effect (and (increase (total-cost) 10)
                      (when (A (is-in ?p) ?pl)
                            (K (is-in ?p)))))
```

4.5. Planning with assumptions

To choose between possible worlds according to the belief state, we introduce assumptive actions. These are virtual actions, that have effects only on **A**-predicates, and which are the only actions that do so.

Definition 2 *(Assumptive action).* An assumptive action $a \in \mathcal{A}^a \subseteq \mathcal{A}$ is an action with the following restrictions:

- In every plan π, a must occur before any physical action.
- All effects in $\text{eff}(a)$ are of the form $\mathbf{A}(v, x), x \in \text{dom}(v)$.
- If a has an effect $\mathbf{A}(v, x)$, then it may never be applied if a conflicting assumptive predicate $\mathbf{A}(v, x'), x' \neq x$ already exists.

The first condition ensures that we can make assumptions only before any real actions are executed. It reflects the fact that assumptions are about the current uncertain state of the world, not future changes. The second condition simply states that assumptive actions cannot affect anything besides assumptive predicates. The third condition prevents the planner from making an assumption that either contradicts a previously made assumption or a known fact about the world (because $v = x$

implies $\mathbf{A}(v, x)$, Definition 1). Thus the planner cannot assume two contradictory facts at the same time. We write the set of assumptions as $\mathcal{A}^a$, and call the remaining actions $\mathcal{A}^p = \mathcal{A} \setminus \mathcal{A}^a$ *physical* actions.

4.5.1. Instance assumptions based on uncertain instance knowledge

Assumptive actions allow us to model probabilistic instance states—for example, in the relational map. If a variable v has a discrete value distribution with $\Pr(v = x_0) = p_0, \ldots, \Pr(v = x_n) = p_n$, we can create a set of assumptions $a_0, \ldots, a_n$ that represent this distribution by setting the effect $\text{eff}(a_i) = \{\mathbf{A}(v, x_i)\}$ and the assumption's probability $\rho(a_i) = p_i$. We use this scheme for closed-form probability distributions over the robot's instance knowledge. Thus, this type of assumptive action provides a solution, in a linear planning framework, to problem (i): planning under uncertainty. Recall the distribution over the room categories for `room1` from the example: $\langle \Pr(\textit{office}) = 0.52,\ \Pr(\textit{corridor}) = 0.24,\ \Pr(\textit{meetingroom}) = 0.24 \rangle$.

In the planning problem, this distribution will be represented by the following assumptions:

```
(:assumption category-room1-corridor
             :effect  (and  (A (category room1) corridor)
                            (assign (probability) 0.24)))
(:assumption category-room1-meetingroom
             :effect  (and  (A (category room1) meetingroom)
                            (assign (probability) 0.24)))
(:assumption category-room1-office
             :effect  (and  (A (category room1) office)
                            (assign (probability) 0.52)))
```

4.5.2. Instance assumptions based on default knowledge

Assumptive actions become much more powerful when combined with default knowledge, providing a solution to open-world planning. They can generate object or place instances with the probability that they exist according to the default-knowledge graph. To achieve this, we specify assumptive actions with parameters for probabilities. The probabilities are filled in at planning time with values obtained from the robot's default knowledge:

```
(:assumptions object-in-room
              :parameters (?l - label ?r - room ?c - category)
              :precondition (and (A (category ?r) ?c)
                                 (not (defined (Pr-contains-instance ?r ?c ?l))))
              :effect (and (A (contains ?r ?l) true)
                           (assign (probability) (Pr-contains-by-default ?c ?l))))
```

states that if we assume a room to be of category `?c`, we assume that an object of type `?l` is in the room with the probability `(Pr-contains-by-default ?c ?l)` that an object of type `?l` exists in a room of category `?c`. For example, the robot can assume the existence of a cornflakes box in a kitchen it has found. These assumptions can be linked in a chain: assume the existence of the kitchen, and then assume the existence of the cornflakes box given that the kitchen exists. If both instance and default knowledge can support an assumption, instance knowledge has priority. Assumptive actions can explicitly prioritize the sources of information employed in their preconditions. In the example above, we disallow the assumption if instance knowledge `(Pr-contains-instance ?r ?c ?l)`, sourced from the relational map, is available. New instance assumptions will also automatically become available to the planner as new places and objects are added to the relational map. Because most planners work on grounded representations of the planning problem, adding new objects during the planning process is not possible. Thus, to enable open-world planning, we create a predefined number of *virtual objects*. Assumptive actions then hypothesize new objects by giving properties to these virtual objects. The number of virtual objects can be scaled as required for a domain.[17] This use of assumptive actions with default knowledge provides our solution to problem (ii): open-world planning.

4.6. Planning in open worlds: an example

In our running example, the following assumptions allow the planner to assume the existence of a meeting room:

```
(:assumption leads-to-room-placeholder1-meetingroom
             :effect (and (A (leads-to-room placeholder1 meetingroom) true)
                          (assign (probability) 0.304)))
```

[17] For example, domains with many unknown rooms and objects would require many virtual objects.

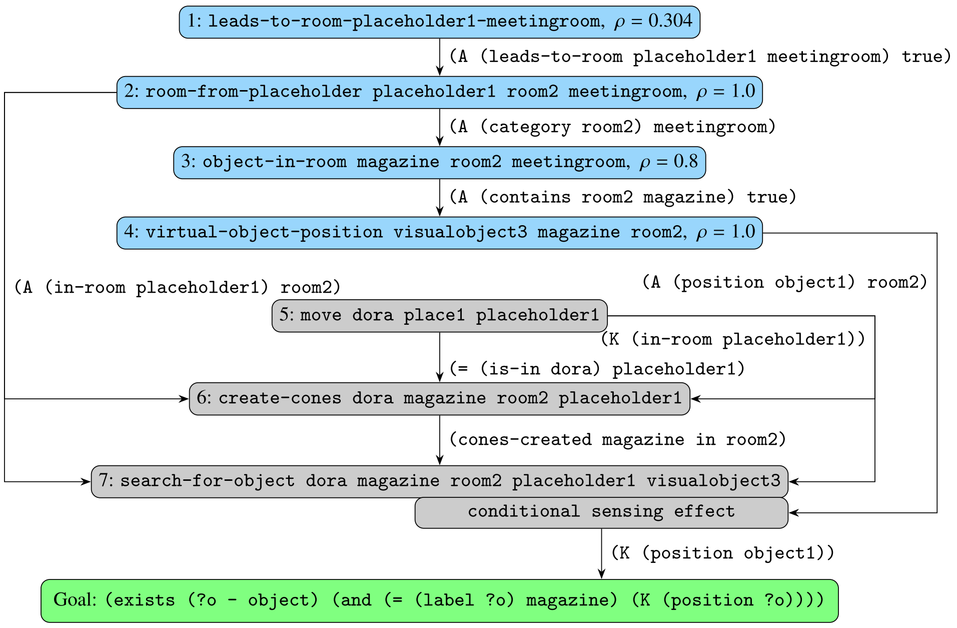

1: leads-to-room-placeholder1-meetingroom, ρ = 0.304
(A (leads-to-room placeholder1 meetingroom) true)
2: room-from-placeholder placeholder1 room2 meetingroom, ρ = 1.0
(A (category room2) meetingroom)
3: object-in-room magazine room2 meetingroom, ρ = 0.8
(A (contains room2 magazine) true)
4: virtual-object-position visualobject3 magazine room2, ρ = 1.0
(A (in-room placeholder1) room2)
(A (position object1) room2)
5: move dora place1 placeholder1
(K (in-room placeholder1))
(= (is-in dora) placeholder1)
6: create-cones dora magazine room2 placeholder1
(cones-created magazine in room2)
7: search-for-object dora magazine room2 placeholder1 visualobject3
conditional sensing effect
(K (position object1))
Goal: (exists (?o - object) (and (= (label ?o) magazine) (K (position ?o))))

Fig. 7. A visualization of the action dependencies of the plan from Fig. 3. Assumptions are coloured blue, and physical actions are coloured grey. Arrows indicate that the first action has an effect which the second action depends on via preconditions or conditional effects. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

```
(:assumption room-from-placeholder
             :parameters (?p - place ?r - room ?c - category)
             :precondition (and (= (placestatus ?p) placeholder)
                                (A (leads-to-room ?p ?c) true)
                                (is-virtual ?r))
             :effect (and (assign (in-room ?p) ?r)
                          (assign (category ?r) ?c)))
```

The first assumption is created from instance knowledge and has a likelihood equal to the probability of finding a meeting room somewhere beyond `placeholder1`. This is computed from the relational map and the default-knowledge graph. Specifically, it uses the likelihoods of room-type connectivity, from known rooms to a room of the desired type. The second assumption uses a virtual room object (in this case `room2`, identified as virtual by the `is-virtual` predicate) to "create" a meeting room that lies beyond the place-holder `placeholder1`.

4.7. *The switching planner: an example*

With these concepts, we can now illustrate how the switching planner in our system works. At the beginning of each planning process we create a plain PDDL planning task that can be used by Fast Downward [18] as follows:

1. Uncertain instance knowledge is converted into grounded assumptions (Section 4.5).[18]
2. Preconditions that enforce the conditions from Definition 2 (assumptions may not be contradictory and must occur before any physical action) are added to all assumptive actions.
3. Observation models are converted into conditional knowledge effects (Section 4.4).
4. Certain instance knowledge facts are assignments in the initial planning state.

All these transformations are inexpensive to perform and are applied every time the deterministic planner is called. Creating a straight-line plan from this planning task will result in the example plan in Fig. 3. To illustrate how the actions interact, Fig. 7 shows the same plan represented as a graph. Note the multiple actions and preconditions that some assumptions support.

If one of the assumptions in that plan turns out to be false, or if the plan execution fails for any other reason, a new plan is created and the process starts again. After several planning-execution cycles, we may reach a physical action with

[18] By contrast, instance assumptions from default knowledge are parameterized, and are grounded with probabilities, internally by the planner.

```
(object-in-room-instance magazine room2 meetingroom) ρ = 0.8

    (virtual-object-position object1 magazine room2) ρ = 1.0
```

(a) The two relevant assumptions for the search action.

```
(:init
  (= (is-in dora) place15)                                  Deterministic facts
  (= (category room2) meetingroom)
  (probabilistic 0.8 (and (= (contains room2 magazine) true)   Probabilistic facts from assumptions
                          (= (position object1) room2))))
```

(b) The grounded minimal belief state induced by the planned assumptions. The single `probabilistic` branch states that the facts `(= (contains room2 magazine) true)` and `(= (position visualobject4) room2))` hold with probability 0.8. The unconditional entropy H of this state is 0.5

```
(:init
  (= (is-in dora) place15)
  (= (category room2) meetingroom)
  (probabilistic 0.8 (and (= (contains room2 magazine) true)
                          (= (position object1) room2)
                          (probabilistic 0.33 (= (visible object1 conegroup0))        H = 0.43
                                         0.39 (= (visible object1 conegroup1))        H = 0.41
                                         0.09 (= (visible object1 conegroup2))        H = 0.48
                                         0.19 (= (visible object1 conegroup3))))))    H = 0.46
```

(c) The extended belief state, after addition of relevant facts. The H values give the conditional entropy of the minimal state in Fig. 8(b) given each additional fact.

Fig. 8. The actions and belief state, taken from the POMDP that is constructed for the problem of a visual search for the magazine in the meeting room.

a non-deterministic outcome.[19] In the second plan in our running example, shown in Fig. 4, the `search-for-object` action's knowledge effect depends on the assumption that the magazine is in `room2`. This *triggering action* causes the planner to switch to decision-theoretic mode [11].

The goal of the POMDP subproblem is to achieve the effect of the triggering action. To ensure the POMDP is small enough to solve, we include only relevant facts in its domain, starting with an absolutely minimal set of variables. These variables are all the facts of which we are certain, and any uncertain assumptive predicates on which the triggering action depends, called *relevant assumptions*.[20] Fig. 8(a) shows the relevant assumptions for our example, and Fig. 8(b) shows the resulting minimal initial belief state. To this minimal set, we add a limited number of uncertain variables, from the full domain, which (a) are observable by the robot, and (b) if known help to determine whether the minimal relevant assumptions hold. If there are many such variables, the belief space will grow too large. We order and select them by the amount of information they give as to whether the minimal relevant assumptions hold. The measure of information used is the *conditional entropy* of the minimal belief state. We first compute the unconditional entropy $H(b_0) = -\sum_{s\in b_0} \Pr(s)\log\Pr(s)$ of the minimal belief state b_0. In our example, this is $-(0.8\log 0.8 + 0.2\log 0.2) = 0.5$. Then for every eligible fact y, we compute the conditional entropy $H(b_0|y) = \Pr(y)H(b_0|y) + \Pr(\neg y)H(b_0|\neg y)$. In our example, we consider the `(visible object1)` facts, which represent whether the object lies within a group of view cones. Fig. 8(c) shows a belief state in which facts about four cone groups have been added, along with their conditional entropies with respect to the minimal state b_0. We select the facts greedily, adding those with the lowest conditional entropy first. We continue until there is no more benefit to be gained from adding new facts, or the size of the belief state is too big. The resulting belief state in our example is quite small, as shown in Fig. 8(c). This formulation of the state space for the POMDP subproblem with use of assumptions contributes to our solution to problems (i) and (ii).

Having formulated the POMDP subproblem, the robot plans for this subtask, executes the policy, and repeatedly updates the belief state. When the belief about the magazine's presence (or absence) has passed a threshold, the POMDP goal action is executed, committing to the presence (absence) of the magazine. Control passes back to the continual planner, which then either continues executing its original plan, or replans.

5. Explaining failures

If the robot declares a task failure ("I can't find a plan likely to find the magazine"), this means that the likelihood of success of the best remaining plan is too low to consider. At this point Dora should explain the failure and verify its explanation—problems (iii) and (iv). This section describes how both problems can be solved using assumptions. The first

[19] A physical action is non-deterministic if it either has an uncertain effect or has a conditional effect that depends on an uncertain assumption.

[20] These are the direct or indirect preconditions of the triggering action.

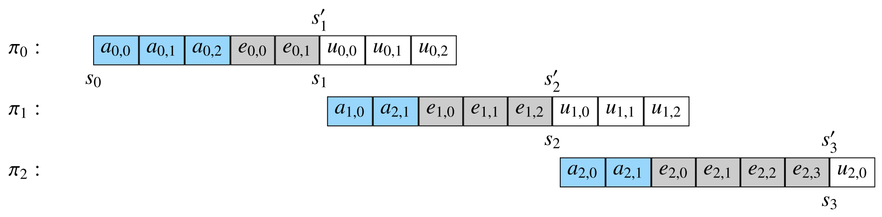

π_0 :
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $e_{0,0}$ $e_{0,1}$ $u_{0,0}$ $u_{0,1}$ $u_{0,2}$
s_0
s'_1
s_1
π_1 :
$a_{1,0}$ $a_{2,1}$ $e_{1,0}$ $e_{1,1}$ $e_{1,2}$ $u_{1,0}$ $u_{1,1}$ $u_{1,2}$
s'_2
s_2
π_2 :
$a_{2,0}$ $a_{2,1}$ $e_{2,0}$ $e_{2,1}$ $e_{2,2}$ $e_{2,3}$ $u_{2,0}$
s'_3
s_3

Fig. 9. A continual planning process consisting of three planning cycles. Assumptions are coloured in blue, and executed actions are coloured in grey. The state resulting from execution of the last action in a plan (e.g. $e_{0,1}$) is the initial state for the next planning cycle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

insight is that a task failure must mean that in each plan so far at least one assumption was incorrect. To formally define what we mean by "explanations" and "failures", however, we cannot just model the last plan, we must model the sequence of plans created by the continual planning process, and thereby the sequence of expected states that results.

Definition 3 *(Continual planning process).* Given a planning task $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, g \rangle$, a *continual planning process* $\mathcal{P}_\Pi = \langle \Pi, \mathcal{L}_\pi, \mathcal{L}_s \rangle$ consists of the underlying planning task Π and sequences of plans $\mathcal{L}_\pi = [\pi_0, \dots, \pi_n]$ and states $\mathcal{L}_s = [s_0, \dots, s_{n+1}]$ with the following properties:

- π_0 is a solution to the original planning task Π; that is, a plan that reaches g from the initial state s_0.
- Each plan π_i can be divided into assumptions, the executed portion of the plan and the unexecuted portion of the plan, $\pi_i^a, \pi_i^e, \pi_i^u$.
- $s_i, i > 0$ is the state observed by the robot after executing π_{i-1}^e.
- π_i is a solution to the modified planning task $\Pi_i = \langle \mathcal{V}, \mathcal{A}, s_i, g \rangle$; that is, a plan that reaches g from the previous observed state s_i.

The *expected state* after plan π_i is denoted $s'_{i+1} = \text{app}(s_i, \pi_i^a, \pi_i^e)$ and is the result of the application of π_i's assumptions and executed actions from its initial state. So, in our example, after searching the meeting room, the robot expects to know where the magazine is. We use the following shorthand for the plan's actions: $a_{i,j}$ for the jth assumption of π_i^a, $e_{i,j}$ for the jth executed action in π_i^e, and $u_{i,j}$ for the jth unexecuted action in π_i^u. See Fig. 9 for an illustration of a sequence of three plans that uses this notation. Given this definition, we define surprise as based on the differences between the expected states s'_i and the perceived states s_i:

Definition 4 *(Surprise).* Let π_i be the ith plan in a continual planning process with initial state s_i, expected state s'_{i+1} after execution of π_i^e, and observed state s_{i+1}. Then a fact σ is a *surprise of* π_i iff $\sigma \in s_{i+1}$ and $\sigma \notin s'_{i+1}$.

So a surprise is a fact in the observed state s_{i+1} that is not part of the expected state s'_{i+1}. Since states are full assignments of values to variables, this includes cases of facts missing from s_{i+1}. In our example, the observed state is not seeing the magazine in the meeting room. This machinery enables us to define the surprises of a continual planning process:

Definition 5 *(Surprises in continual planning).* Let $\mathcal{P}_\Pi$ be a continual planning process. Then the *surprises* $\mathcal{S}$ of $\mathcal{P}_\Pi$ are defined as a sequence of sets of surprises, one for each planning cycle:

$$\mathcal{S} = [\mathcal{S}_0, \dots, \mathcal{S}_n]$$
$$\mathcal{S}_i = \{\sigma : \sigma \text{ is a surprise of } \pi_i\}$$

This definition of surprise is quite general: any observed, but unexpected fact is a surprise. So Dora might be surprised to find a cornflakes box in the office, but this is not helpful in explaining why the magazine was not found. We thus want to concentrate on the subset of surprises that might have caused the task failure: the *task-relevant* surprises. This is the set that breaks at least one precondition of an action later in the plan. In fact, whenever the continual planner replans, this must have been caused by a surprise according to Definition 4. Having defined surprises, we can consider how to explain them, thus addressing our problem number (iv). To do this, we reuse our assumption mechanism. An explanation is defined as a sequence of assumptions that change the expected behaviour to be the same as the behaviour that has been observed.

Definition 6 *(Explanation).* Given a surprise σ of π_i, an *explanation* $\mathcal{E}$ for σ is a sequence of assumptive actions so that

$$\sigma \in \text{app}(s_0, \mathcal{E}, \pi_0^e, \dots, \pi_i^e) \qquad \text{and}$$
$$\text{def}(\text{app}(s_0, \mathcal{E}, \pi_0^e, \dots, \pi_j^e)) \subseteq s_{j+1} \setminus \mathcal{S}_j \qquad 0 < j \le n.$$

$\pi_0^a = \begin{cases}\texttt{category-room2-meetingroom}\\ \texttt{object-in-room magazine room2 meetingroom}\\ \texttt{virtual-object-position object1 magazine room2}\end{cases}$	Assume room 2 is a meeting room which contains the magazine.
$\pi_0^e = \texttt{search-for-object magazine room2 p3 object1}$	Search for the magazine in the meeting room.
$s_1' = \{\texttt{(K (position object1))}\}$	Expected state: the magazine's location is known.
$s_1 = \{\neg\texttt{(K (position object1))}\}$	Observed state: the magazine's location is not known.
$\pi_1^a = \begin{cases}\texttt{category-room1-office}\\ \texttt{object-in-room magazine room1 meetingroom}\\ \texttt{virtual-object-position object1 magazine room1}\end{cases}$	Assume room1 is an office which contains the magazine.
$\pi_1^e = \texttt{search-for-object magazine room1 p7 object1}$	Search for the magazine in the office.
$s_2' = \{\texttt{(K (position object1))}\}$	Expected state: the magazine's location is known.
$s_2 = \{\neg\texttt{(K (position object1))}\}$	Observed state: the magazine's location is not known.

Fig. 10. Extract of information from two partially executed plans from the running example. Both plans fail. The expected and actual states are shown. The task relevant surprises are the two missing knowledge effects.

Given a sequence of sets of surprises $\mathcal{S} = [\mathcal{S}_0, \ldots \mathcal{S}_n]$, $\mathcal{E}$ is an explanation for $\mathcal{S}$ iff it is an explanation for each $\sigma \in \mathcal{S}_i$ for every $\mathcal{S}_i \in \mathcal{S}$.

The first condition simply states that when the original plan sequence is executed, given the explanatory assumptions, the resulting state must contain the surprising observation that we want to explain. The second condition ensures that these explanations do not cause any new surprises. As explanations consist of assumptive actions which each have a probability, the probability of an explanation is simply the product of those probabilities: $\rho(\mathcal{E}) = \prod_{\epsilon \in \mathcal{E}} p(\epsilon)$.

For illustration (Fig. 10), let us take a simplified version of our example task, focusing only on the sensing actions. In this example, we have a sequence of two plans, both failing to achieve their goal. Both plans start with three assumptive actions, first establishing the category of a room, then assuming the existence of the type of target object in that room, and finally placing a virtual object there, so that there is a concrete instance of "magazine" that the planner can reason about. These assumptions are followed by a single `search_for_object` action, which would satisfy the goal `(K (position object1))` if the assumptions were correct.

The surprises are, in both cases, the missing effect of the search action, so $\mathcal{S}_0 = \mathcal{S}_1 = \{\neg\texttt{(K (position object1))}\}$ and $\mathcal{S} = [\mathcal{S}_0, \mathcal{S}_1]$. A possible explanation is that the magazine is actually in a third room, `room3`:

$$\mathcal{E} = [\texttt{category-room3-meetingroom}, \texttt{obj-in-room magazine room3 meetingroom}, \\ \texttt{virtual-object-positionobject1 magazine room3}]$$

would be an explanation for $\mathcal{S}_0$ and $\mathcal{S}_1$ and therefore also for $\mathcal{S}$.

5.1. *Preventing explanation by omission*

This example, however, betrays a weakness of this definition of explanations: it is sufficient to withdraw an assumption to explain a surprise. A sufficient explanation would be $\mathcal{E} = [\texttt{category-room3-meetingroom}]$. Intuitively the explanation is inadequate. The reason lies in the fact that state variables either take a specific value[21] or the unknown value $\perp$. The role of assumptions is precisely to assign specific values to unknown variables. The purpose of those assignments is to enable later physical actions in the plan to succeed, by satisfying the conditions in their desired conditional effects. From our example, consider the action that searches the room for the magazine. Its effect is for Dora to know where the object is. The condition of that effect is that the magazine is in the room. In planning terms, that condition is unsatisfied if the location of the magazine is unknown. Making the assumption that the magazine is in `room2` satisfies the condition. Thus, to trivially explain why Dora did not see the magazine we could simply withdraw the assumption: the location becomes unknown again, the plan will fail, and so an explanation has been provided.

This style of explanation by omission is unsatisfactory. It relies on the implicit closed-world assumption in the action models—everything the robot does not know to be true is assumed to be false. Plan failure is then explained in terms of *the robot not knowing the things needed for the plan to succeed.* But the true state of the world, rather than the robot's lack of knowledge, was the direct cause of plan failure. An explanation must thus commit to specific statements about the world (assignments of specific values to variables) that explain the failure. So this is what is enforced in our approach. We could

[21] By "specific value" we mean any value that is not the unknown assignment $\perp$.

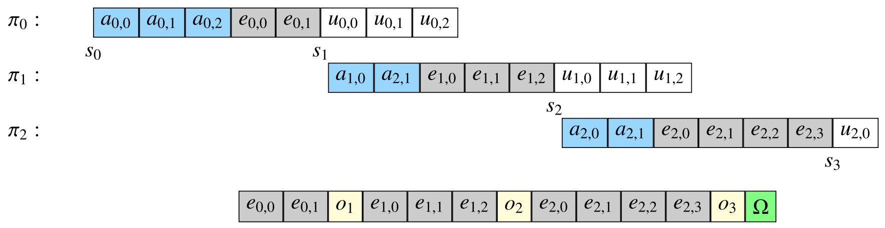

π_0 :
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $e_{0,0}$ $e_{0,1}$ $u_{0,0}$ $u_{0,1}$ $u_{0,2}$
s_0
s_1
π_1 :
$a_{1,0}$ $a_{2,1}$ $e_{1,0}$ $e_{1,1}$ $e_{1,2}$ $u_{1,0}$ $u_{1,1}$ $u_{1,2}$
s_2
π_2 :
$a_{2,0}$ $a_{2,1}$ $e_{2,0}$ $e_{2,1}$ $e_{2,2}$ $e_{2,3}$ $u_{2,0}$
s_3
$e_{0,0}$ $e_{0,1}$ o_1 $e_{1,0}$ $e_{1,1}$ $e_{1,2}$ o_2 $e_{2,0}$ $e_{2,1}$ $e_{2,2}$ $e_{2,3}$ o_3 Ω

Fig. 11. The action sequence (bottom row) enforced on the planner via the action-ordering variable M, so as to reproduce the executed portions of three, partially executed, plans (top three rows). Each observation action o_i requires that the immediately previous physical action resulted in the corresponding observed states s_i.

do this by assuming a value for every single variable in the state description. In other words, to construct an explanation by completely defining a possible world. This is unnecessary. Since the robot only needs to explain the task-relevant surprises, it needs to fix the values of only those variables that might have affected the outcomes of its plan. A minimal set of variables is thus those which were initially unknown, and that occur in the conditions of conditional effects for the executed actions. No other variables, which can be changed by assumptions, could influence any of the outcomes that caused failure. Enforcing an assignment, with use of assumptions, to every variable in this minimal set, ensures that the explanation produced *cannot* rely on explanation by omission. Of course, this minimal set may not be enough, so assumptions about additional variables are allowed. Formally, we require that if $e \in \bigcup_{i=0}^{n} \pi_i^e$ was an executed action and $C \triangleright \mathit{eff}$ is a conditional effect of e, then for each condition $v_i = x_i \in C$ there must be an assumption $\epsilon \in \mathcal{E}$ and a value $x' \in \mathrm{dom}(v_i)$ with $v_i = x' \in \mathrm{eff}(\epsilon)$. In the running example, the conditions in the search actions' conditional effects would force us to make some assumption about the position of the magazine—that is, about `(position object1)`.

5.2. Finding explanations as a planning problem

Since an explanation is a sequence of assumptions, finding one is a new planning problem. We allow the planner to create a new sequence of assumptions that precedes the executed actions of the original sequence of plans. Then we force the planner to replay that sequence of executed actions, checking that the physical actions now cause the effects originally observed by the robot. If a plan is found, its assumptions are an explanation for the task-relevant surprises.

Given a continual planning process $\mathcal{P}_\Pi$ with $\Pi = \langle \mathcal{V}, \mathcal{A}, s_0, g \rangle$, we create a new planning task $\hat{\Pi} = \langle \hat{\mathcal{V}}, \hat{\mathcal{A}}, \hat{s}_0, \hat{g} \rangle$ as follows:

$$\hat{\mathcal{A}}^a = \mathcal{A}^a$$ The set of possible assumptions is the same.

$$\hat{\mathcal{A}}^p = \{o_1, \ldots, o_{n+1}\} \cup \pi_0^e, \ldots, \pi_n^e$$ We add special observation actions.

$$\mathrm{pre}(o_i) = s_{i+1}, \ \ \mathrm{eff}(o_i) = \emptyset$$ An observation action checks that the expected state occurs at the end of a plan.

$$\hat{\mathcal{V}} = \mathcal{V} \cup \{M\} \text{ with } \mathrm{dom}(M) = \hat{\mathcal{A}}^p \cup \{\Omega\}$$ We add names for the physical actions and a special termination variable to the state.

$$\hat{s_0} = s_0 \cup \{M = e_{0,0}\}$$ We add the name for the first physical action to the initial state.

$$\hat{g} = (M = \Omega)$$ The goal is to reach the termination action—that is, to execute the entire original action sequence.

The planner is allowed to make all the assumptions $\mathcal{A}^a$ of the original plan, but the only physical actions that are allowed are the executed actions of the original plan. To these we add a set of *observation actions* o_i, which are used to check if the simulated execution results in the same state sequence as that originally experienced by the robot. We also add an action variable M to the state. This forces the planner to execute all physical actions in order (see Fig. 11). In the initial state it has the value of the first executed action $e_{0,0}$, and the goal is to reach the final value Ω, i.e. to execute all physical actions. To that end, every physical or observation action $a \in \hat{\mathcal{A}}^p$ has a precondition $M = a$ and an effect $M = \mathrm{succ}(a)$, with the successor function $\mathrm{succ}(\cdot)$ defined as follows:

$$\mathrm{succ}(e_{i,j}) = \begin{cases} o_{i+1} & \text{if } j+1 = |\pi_i^e| \\ e_{i,j+1} & \text{else} \end{cases} \qquad \mathrm{succ}(o_i) = \begin{cases} \Omega & \text{if } i > n \\ e_{i,0} & \text{else} \end{cases}$$

Recall that assumptions may occur only before the first physical action. Thus, the planning problem has a solution only if the planner can find a sequence of assumptions that allow all the original physical actions to be executed in sequence, and which results in expected and actual observations s_i matching. If we want to force the planner to make assumptions about certain variables $v \in \mathcal{V}$ to prevent explanations by omission, we can add the required conditions to the goal. This approach finds explanations, but has two shortcomings:

1. *It tries to explain surprises it cannot explain.* Every surprise must be explained. So if an external agent changes a fact about the environment that the robot implicitly assumes to be static, no explanations will be found, not even for surprises that could be explained. We would like partial explanations when full explanations are not possible.
2. *It will make more assumptions than necessary.* Even if every surprise can be explained, the planner may have to make a large number of assumptions, even though most surprises can be explained in a straightforward way. This can cause a substantial slowdown if longer action sequences have to be explained.

The solution to the first problem is to avoid trying to explain the unexplainable. This is achieved by limiting the facts checked by the observation actions to those that are caused by conditional effects—that is, to those observations that can be affected by assumptions. We note that for a variable v and the observed states s_i and s_{i+1}, there are three possibilities for the actions π_i^e that were executed in between:

- **Case 1:** v occurs only in unconditional effects of π_i^e. Then v will change as expected, or it has a cause outside the domain model. So it cannot be explained.
- **Case 2:** v does not occur in any effect of π_i^e. Then any change in v will have a cause outside the domain model. So, again, it cannot be explained.
- **Case 3:** v occurs in at least one conditional effect of π_i^e. A change in v may be caused by the conditional effect triggering differently from expected, which could be changed by an assumption. Thus, it might be explainable.

Hence the observation action o_{i+1} needs to check observations only in case 3; in the first two cases the value is either expected or inexplicable. Unchecked facts are instead part of the effect of o_{i+1}, so the remaining plan is simulated correctly.

To address the second problem, we use the fact that assumptions can often be determined in an efficient backward chaining manner, rather than by forward search. Specifically, many assumptions can be determined from one observation. If an observation is unexpected, and all the conditional effects that cause it have a common subset of conditions, then it follows that this common subset must hold. Thus, assumptions that set the conditions must be part of any explanation. Formally, if a variable v changes between s_i and s_{i+1} from x to x', then let $C = \{c_0, \dots, c_m\}$ denote the conditions of those conditional effects $c_j \triangleright e_j$, which assign $v = x'$ as an effect—that is, $(v = x') \in e_j$. If there is a set of common preconditions $\text{Pre}_v = \{\mathbf{A}(v_0, x_0), \dots, \mathbf{A}(v_n, x_n)\}$ so that $\text{Pre}_v \subseteq c_i$ for all conditions $c_i \in C$, then the assumptions in Pre_v must be contained in any explanation. In practice this improvement significantly cuts planning time, especially for longer failed plans. It also leads to shorter explanations.

We can force the planner to return multiple explanations. Each is an explanation ranked by its probability according to the belief state, thereby allowing the robot to verify explanations in order of plausibility. In this assumptive framework, failures can be explained with the use of assumptions based on either instance or default knowledge. In the first case, failures are explained by the choosing of uncertain facts from the relational map—such as positing that the type of room searched was different from that originally supposed. In the second case, by our allowing instance assumptions based on default knowledge, explanations posit the existence of new places or objects—such as the existence of a new room the object might be in. Thus, by using assumptions for explanations, we can also leverage the mix of relational and commonsense knowledge employed for regular task planning. We refer to these as closed-world and open-world explanations. However, without additional machinery these explanations are still quite limited in power. To give them more power we need to introduce the third level of our knowledge schema: diagnostic knowledge.

5.3. Diagnostic knowledge

Explaining failures requires explicit assumptions, but many assumptions in the default layer are implicit. To find richer explanations we must make these implicit assumptions explicit. The diagnostic layer contains two types of action models that achieve this: *diagnostic actions* and *default assumptions*.

Diagnostic actions are based on the default layer's models of physical actions, augmented with additional possible outcomes and their causes. As an example, recall the default `move` action from Fig. 12(a). This has two effects: one physical effect, in which the robot ends up at the new location; and one knowledge effect, in which the robot knows the room type of the new location. But even if this action's preconditions are satisfied, the effects are not guaranteed. The route might be temporarily blocked by a person, for example, or the door might be closed. In this case the physical effect will not happen. Even if the robot can reach the new location, the knowledge effect may fail: if the place node represents a doorway, there *is* no room to which it belongs. We can explicitly model these failure modes using conditional effects. Fig. 12(b) shows this for the `move` action: the preconditions for the action's execution remain the same but its effects are dependent on additional predicates. This augmented action model is stored in the diagnostic layer. In fact every physical action model in the default layer has a more complex counterpart in the diagnostic layer. When explanations are required, these more complex action models are recruited. This gives the planner many more possible failure modes, expressed using conditional effects, and thus a greater variety of explanations.

The second type of diagnostic knowledge is a new kind of assumption. The default layer contains assumptive actions that hypothesize new instance states. In the diagnostic layer there are corresponding *default assumptions* that can hypothesize new default knowledge about state. This default state knowledge includes `is-a` and `contains` relations between

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

(a) Original version

```
(:action move
 :parameters (?t - robot ?from ?to - place)
 :precondition (and (connected ?from ?to)
                    (= (is-in ?a) ?from))
 :effect (and
          (when (and (A (place-exists ?to) true)
                     (A (gateway ?to) false)
                     (A (blocked ?from ?to) false))
                (K (in-room ?to)))
          (when (and (A (place-exists ?to) true)
                     (A (blocked ?from ?to) false))
                (and (assign (is-in ?a) ?to)))))
```

(b) Augmented version

Fig. 12. The default model of the `move` action from the default layer's planning domain and the augmented diagnostic action model from the diagnostic layer.

types of entities. A default assumptive action has as its effect an assumption that a new relation exists between two entities already in the default-knowledge graph. For example, `(A (contains (meetingroom container)))`, meaning "meeting rooms typically have containers". Since default knowledge affects the inferences made about relational state, new defaults increase the likelihoods of explanations involving instance assumptions. Thus, default assumptions allow the robot to generate more general explanations than are possible with instance assumptions alone. For example, only with default assumptions can the robot hypothesize that cereal boxes are often kept in cupboards in dining rooms.

The diagnostic actions and default assumptive actions are employed only for planning of explanations. This is because they would add unnecessary complexity to regular task planning. It is also an open question how to assign probabilities to default assumptions, as they deal by definition with new knowledge. Diagnostic actions are restricted to planning of explanations because their many conditional effects slow regular task planning.

5.4. Verification

If an explanation has been found, the planner can pass the explanation to the motivation component, which can then decide to try to verify whether the explanation is correct. Verification tasks are normal planning tasks, which use the default layer's planning domain. Typically the only way to verify explanations is to ask questions of a human, since if other options were available, they were exhausted during the initial round of task planning. This, in turn, can require assumptions about the locations of humans able to answer the robot's questions. However, the robot cannot currently confirm all explanations it can produce. In particular, assumptions about the *non*-existence of objects are things the robot cannot plan to verify. Since planning to verify assumptions also uses our assumptive framework, this completes our approach to problem (iv).

6. Implementation

In this section, the robot implementation is described. We start with the architecture, and follow with brief descriptions of the components for localization and mapping, place categorization, navigation, object search, person search, dialogue, and the sourcing of default knowledge.

6.1. Architecture

We use a three-layer architecture, comparable in some respects to others such as 3T [32]. The three layers of the architecture are independent of the three knowledge layers. The architectural layers, shown in Fig. 2, are termed the *competence*, *belief*, and *deliberative* layers. The competence layer contains concurrently running sensor and action processes. Sensory competences send information to the belief layer. Action competences implement the physical actions of the planner. The deliberative layer contains the planner, and systems for execution and goal management. The belief layer mediates between the competence and deliberative layers. It contains the relational map, the default-knowledge graph, and the chain graph inference engine that maintains the belief state. See Sections 1 and 3) for further information.

The function of the belief layer is to provide a stable set of symbols for planning. Stability means that a symbol must persist at least as long as the planner takes to use it. This requires careful symbol design, abstracting away rapidly changing information that is irrelevant to task planning. Belief state updates must be timely, and incremental, as signals arrive asynchronously from the different competences. The competence layer provides the belief layer with rapidly changing instance information from each modality (dialogue, depth sensing, vision, and laser). Each competence maintains its own internal representation (e.g. a metric map), and is itself composed of multiple components that asynchronously update a competence-specific working memory, and transmit abstract information only to the knowledge layer [52]. Components in the competence layer either run continuously or are explicitly called on demand as needed. Examples of the former are metric mapping, localization, visual place appearance classification and topological map maintenance. Examples of the latter

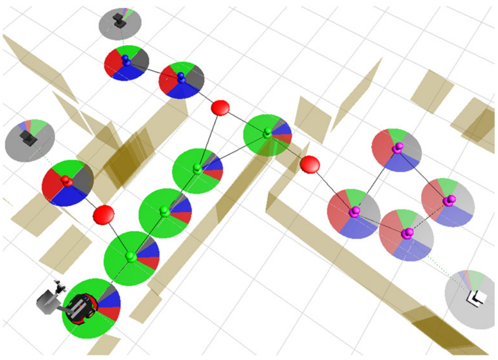


Fig. 13. A place map with several places and three detected doors shown as red dots. Colours on circular discs indicate the probability of room categories as in a pie chart. Here green is *corridor*, red is *kitchen* and blue is *office*. Grey circles denote place-holders in unexplored space. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

are natural language processing, and active search behaviours for objects and humans. Components are called on demand if they are resource intensive, or only needed occasionally.

In summary, the belief layer implements a representational separation between the deliberative and competence layers. It also insulates the deliberative layer from implementation changes to components in the competence layer. The entire architecture is an instantiation of the CoSy Architecture Schema (CAS) [6], implemented with the CAS Toolkit (CAST) [16]. This is a component-based, event-driven integration framework that has been used to develop multiple robot systems [45, 15,13]. CAST enables development of loosely coupled, component-based systems governed by the exchange of information and knowledge via multiple *working memories*. We describe each competence in turn.

6.2. Continuously operating competences

Continuously operating competences run constantly and push information to the belief layer, causing asynchronous updates to the relational map. They also maintain their own local representations—for example, a metric map is maintained by the mapping competence—which are not exposed to the belief layer explicitly, but which are used by other competences (e.g. navigation and movement).

6.2.1. Localization and mapping

We employ a hybrid map representation, comprising feature-based and local metric grid maps, together with a topological graph. This topological map consists of nodes called *places* (corresponding one to one with the places in the relational map), connected through edges representing traversable routes. The underlying simultaneous localization and mapping techniques and obstacle perception methods are based on work by Hawes et al. [15], Pronobis et al. [37] and Zender et al. [55]. For the purpose of this paper, it is enough to know that these techniques allow the robot to be well localized and to build a 3D metric map using information taken from 2D laser scans, 3D point clouds from a depth camera, and odometry data. This metric map explicitly represents free space at floor level and the heights of detected objects, and removes transient obstacles after repeated scans. None of this metric information is transmitted to the knowledge layer, but is used for navigation when movement actions are executed. Places are created at regular intervals as the robot moves, and at doorways. Doorways are detected in the laser scan, and rooms are defined as groups of nodes, each room being connected to another room by at least one doorway [15]. Doorways are defined as not belonging to any room. Space that has not yet been traversed by the robot has no places in it. Therefore, the robot can add hypothesized places in unexplored locations within 2D laser scan range. These hypothetical places allow reasoning about unvisited space, and planning and execution of exploration. Hypothetical places correspond to the *place-holders* in the relational map. For an example map, see Fig. 13.

6.2.2. Place categorization

A categorization algorithm augments place nodes with information about room categories as described by Pronobis et al. [36]. Following Pronobis and Jensfelt [35], a small set of views is acquired at each place, and two types of low-level features are extracted: (a) geometric features from laser range data which are indicative of room shape and size, and (b) global appearance features based on composed receptive field histograms obtained from second-order, normalized Gaussian derivative filters applied to the illumination channel, captured from a wide-angle camera, at two scales. On the basis of these features, independent categorical models are built for shape (e.g. elongated or rectangular), size (e.g. small or large) and visual appearance (e.g corridor-like or office-like) properties, with classification being realized with support vector machines. For each of the places, the categorizer then creates relations between that place and properties—for example, a place is linked to the geometric concept “rectangular”. These place-property relations are modelled probabilistically, on the basis of confidence measures derived from the distances between the classified samples and discriminative model

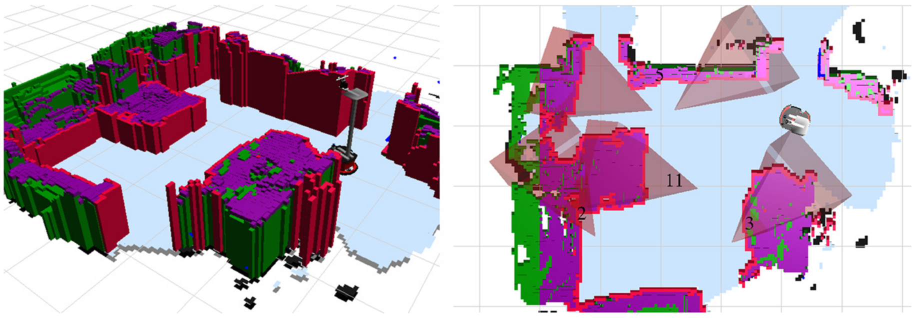


Fig. 14. Creating view-point actions for object search. **(Left)** The prior probability distribution for an object in a room (purple), based on the locations of likely supporting surfaces (green). **(Right)** Examples of view cones with probabilities. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

hyperplanes [36]. The accumulated confidences gained from the support vector machine models across views are normalized to gain probabilities. The place-property relations are transmitted with their probabilities to the belief layer.

6.3. *On-demand competences*

On-demand competences run only when explicitly triggered by an action in a plan devised by the deliberative layer in the architecture. All such actions have epistemic effects, as modelled in the default-knowledge graph.

6.3.1. *Navigation*

The movement and navigation competence implements the move actions available to the planner. Movement can occur between pairs of adjacent or non-adjacent places. In the latter case, the robot does not have to travel exactly through intermediate place nodes in the topological graph. The `move` action manoeuvres the robot to the centre of a place. Precise navigation is performed by communication with the localization and mapping competence described above. That competence also estimates the place node of the robot at the end of the movement and communicates it to the belief layer.

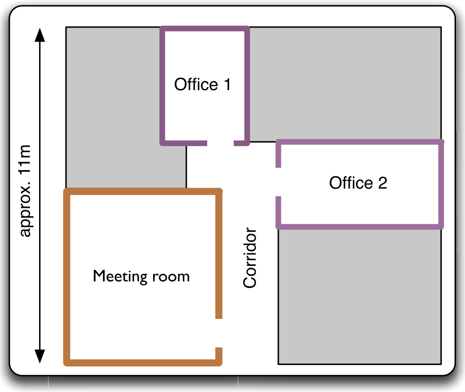
6.3.2. *Object search*

To find objects, the robot must conduct a *visual search*. Visual object detection uses scale-invariant feature transform (SIFT) based recognition [40] of pretrained object instance models. The problem of *active visual search* has previously been investigated, e.g. by Kollar and Roy [24], Aydemir et al. [1], Ekvall et al. [7]. We employ an approach developed by [3], utilizing topological relations proposed by Sjöö et al. [44]. This works as follows. The 3D grid map and target object type jointly determine a probability distribution over the object location—for example, many objects (cornflakes boxes, books, magazines) are often located on raised surfaces such as tables. Our view-point generation component exploits this information, using a strategy similar to that in [12]. The object probability distribution comprises a set of uniform distributions on the real plane, one for each likely supporting surface, as shown in Fig. 14. Places in the topological map constitute the possible viewing locations, and different viewing directions are sampled from each place. Each potential view point is then evaluated on the basis of the probability of the target object being within the view cone defined by the view point, taking into account the field of view of the camera. A set of view points is grown in a greedy fashion until at least 95% of the probability mass is covered (Fig. 14). The view points sharing a place are grouped, those groups are transmitted to the belief layer, and are added to the relational map. The switching planner then uses them in planning visual search.

6.3.3. *Dialogue*

The system engages in dialogue, both to receive goals and to acquire knowledge from a human. A dialogue initiated by the robot is very similar to any other planned knowledge-acquiring action and hence is modelled similarly. The dialogue model is based on the work of Skočaj et al. [46], Janíček [21] and will only briefly be outlined in the following. There are two types of dialogue actions: *engagement*, and *questions* triggered by the deliberative layer. The answers to questions yield an update to the relational map. We support two question types: polar questions[22] ("Is this a kitchen?") and open questions ("What room is this?"). In the planner, polar questions are modelled as more reliable since they have a lower risk of being misinterpreted. The question content is generated by reference to the relational map and the default-knowledge graph. We employ the approach by Zender et al. [54] to generate such questions, and in particular the generate suitable referring expressions. For the experiments reported here, natural language input was through text rather than speech recognition, but the rest of the natural language processing chain was employed. The robot generated spoken output.

[22] By "polar question" we simply mean a question that can be answered with one of two options, e.g. yes or no.



(a) The test environment.

	Explore	Categorize	Find
Total planning time	115.7 s	135.6 s	301.9 s
Time per plan call	5.0 s	17.8 s	15.4 s

(b) Summary of planning time for all runs in the three different test conditions (secs).

Exp.	Task	Condition	No. of runs	No. of successful runs	
1	Explore	C_{laser}	3	3	(100%)
		$C_{combined}$	4	4	(100%)
		Total	**7**	**7**	**(100%)**
2	Categorize	C_{laser}	3	3	(100%)
		$C_{combined}$	3	3	(100%)
		Total	**6**	**6**	**(100%)**
3	Find	C_c	14	7	(50%)
		C_{nc}	7	4	(43%)
		Total	**21**	**11**	**(52%)**
4	Explain	N	7	6	(86%)
5	Verify	N	2	1	(50%)
		Total	**9**	**7**	**(78%)**
	Grand total		**46**	31	**(67%)**

(c) Summary of all runs. Note that the figures reported here refer to the total number of individual runs explicitly designed to be tested in each test condition (determined by the task and the configuration), along with the number of successful runs in each of these test conditions.

Fig. 15. Environment and summary of runs.

6.3.4. *Person search*

Finding a person to talk to is a precondition for dialogue. We implemented an active person search competence based on RGBD images. It is modelled by the planner as an action `look-for-person` that has the uncertain effect of seeing a person at a place when that person is within 3 m. Whenever the action is called, the robot scans the room by rotating through 360°. It stops between scans, in angles of 80% of the aperture of the depth camera, so as to provide some overlap between views. For each view acquired, the OpenCV face detector (using cascades of simple detectors for simple Haar-like features [28]) is run to yield a number of potential faces. To reduce the number of false positives, those hypotheses are filtered with use of a simple heuristic: the distance to a face—estimated by our assuming a constant size for human faces—must be within 50 cm of the median depth camera output for the corresponding pixels. The detected person is associated with the nearest place in the relational map.

6.4. *Acquiring probabilistic knowledge*

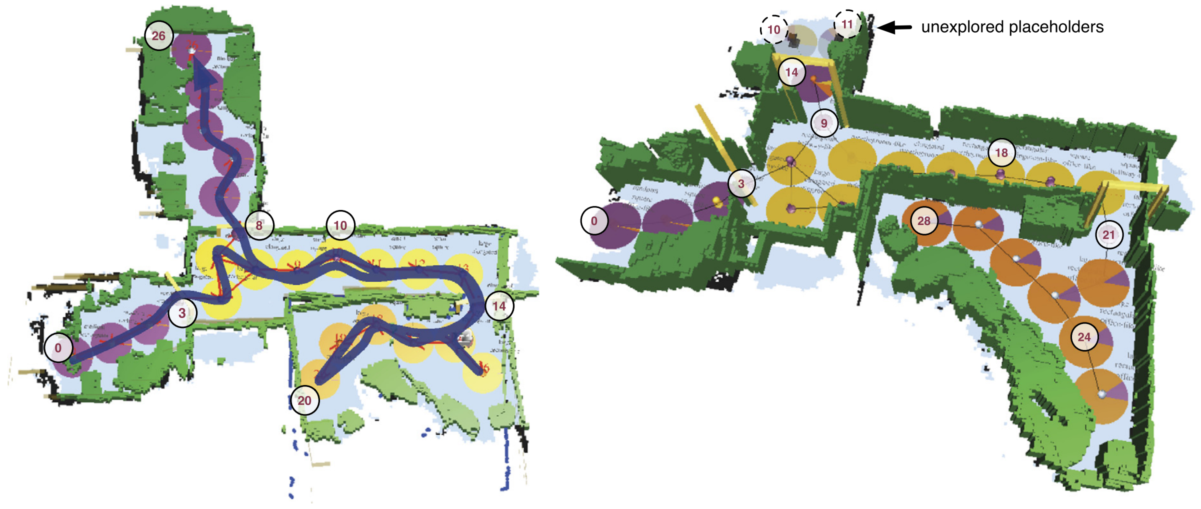
While the structure of the default knowledge can be designed manually, we require probabilities for the relations encoded. These are captured semi-automatically for quantities such as object type-room type relations. A set of possible object-room relations was extracted from the Open Mind Indoor Common Sense (OMICS) database.[23] Probabilities for relations were estimated from image search-engine queries on the Web [13]. The observation models for the POMDP planner were built empirically. We ran object detection on a range of target objects, in two office environments, and under varying lighting conditions. Room categorization reliabilities were calculated using confidence scores from learned prototypes, as described in Section 6.2.2.

7. Evaluation

This section reports the results of 46 test runs, all made in an office environment depicted in Fig. 15(a). Three different tasks were given: explore space, categorize rooms and find a target object. The explanation subsystem was further tested in two additional experiments, as shown in Fig. 15(c). The purpose of these experiments is to show the solution to the four problems outlined: (i) planning under state and observation uncertainty, (ii) planning in open worlds with incomplete information, (iii) planning to explain failure and (iv) planning to verify explanations.

In *experiment 1*, the robot was tasked with *exploring* with the goal of building a complete topological map. While doing this, the robot passively senses room categories, using either the laser or both the laser and visual appearance (Section 6.2.2). In *experiment 2*, Dora was given the goal to *categorize* all the rooms in the environment, allowing active sensing in the form of dialogue. This experiment shows the ability of the planner to deploy the best information-gathering modality. It also shows the use of instance assumptions from instance knowledge to plan under uncertainty. In *experiment 3*, the robot was tasked, as in the running example, with *finding* an object, starting with no map. This experiment shows the ability to switch to decision-theoretic sessions, and to use instance assumptions derived from default knowledge, to plan in open worlds. In

[23] See http://openmind.hri-us.com/. This was obtained from humans freely listing object types they associated with given room types.



(a) Exploration and categorization. The route taken by the robot during exploration is superimposed on the map in blue (see Figure 17). The probabilities of room categories, shown by the pie charts for each place, result from the sequence of actions shown in Figure 18.

(b) Final map during an object search run. The robot started in place 0 and found the object in place 24.

Fig. 16. The robot's maps at the end of the runs described in Sections 7.2 and 7.4. The offices are located on the left and at the top left. Three-dimensional obstacles are shown in green. The pie charts show room category probabilities. The category colours are as follows: purple for *office*, yellow for *corridor* and orange for *meetingroom*. Places are numbered, corresponding to the course of action plots in Figs. 17, 18 and 20. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

experiment 4, all the suitable runs that failed in experiments 1–3, plus seven additional runs with forced failures, were given to the planner to explain the failures using diagnostic knowledge. This experiment shows the ability of the system to use a mix of default assumptions and instance assumptions to create explanations for task failure. Finally, in *experiment 5*, the ability to verify the explanation was tested, as in the running example. This shows the ability to plan again with instance assumptions in order to verify an explanation that involves hypothesized new default knowledge.

7.1. Overall quantitative analysis

The runs for each test condition[24] are summarized in Fig. 15(c). The system exhibits robust behaviour during *exploration* and *categorization*, but only succeeds half the time for *object search*. Explanations were found in most cases, but only one of the two attempts at verification of an explanation fully succeeded. Success criteria were strict: no intervention by the operator was allowed, and if the robot was physically inactive for more than five minutes, the run was cancelled and recorded as a timeout failure. In Fig. 15(b) the time spent on planning under the different test conditions is reported. The absolute planning time integrates the duration of all planner calls in a run and averages them over all successful runs for each test condition. The absolute planning time is significantly higher for the *find* test condition compared to *categorize* and *explore*. This indicates the inherently more complicated planning problem in the *find* test condition. Section 7.4 sheds light on failure modes. All the analyses record the overall runtime, the state uncertainty, and the action sequences taken by the robot.

7.2. Experiment 1—explore: continual planning

The robot was tasked with exploring all of its accessible environment, using the following goal:

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

thereby building a complete topological map. While doing this, room-categorization routines run passively in the background. To show the effect of different sensing modalities on categorization ability (Section 6.2.2), we used two different test conditions: test condition C_{laser}, where only laser-based features were used for room categorization; and test condition $C_{combined}$, where the robot had both laser and visual appearance features. This should not affect exploration behaviour, but should affect room categorization ability. Three runs were made for C_{laser} and four runs were made for $C_{combined}$, each run starting in a different room. Each run started without a map.

An example exploration run under test condition C_{laser} starting in Office 1 (place 0) is shown in Figs. 16(a) and 17. In Fig. 16(a) the final map is shown with all place-holders and rooms explored. The executed action sequence is shown in

[24] A test condition is a combination of task, robot configuration, and environment configuration.

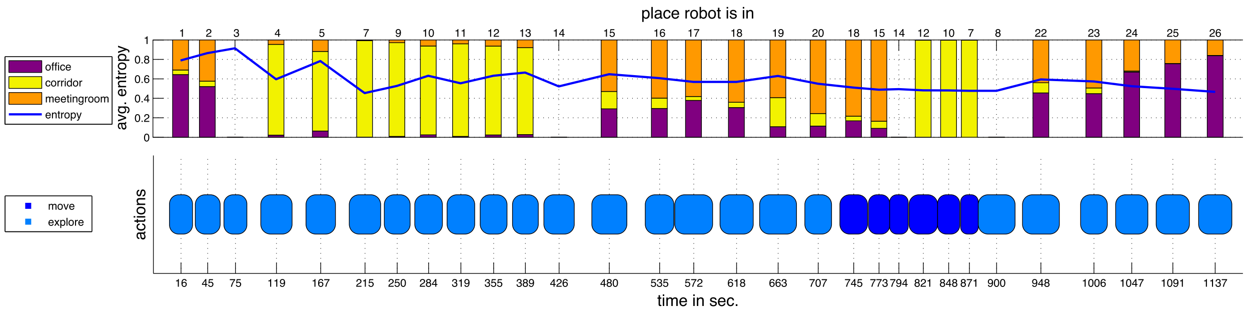


Fig. 17. The sequence of the robot's beliefs (top) and actions (bottom) during an exploration run. The top plot shows the room category distribution after each action. The blue line gives the entropy of the room category distributions, averaged over all rooms found. The numbers at the top refer to the robot's location, corresponding to the annotations in Fig. 16(a). The bottom plot shows the action sequence, indicating execution durations. The action type is colour-coded. Blanks indicate that the robot was planning for that period. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

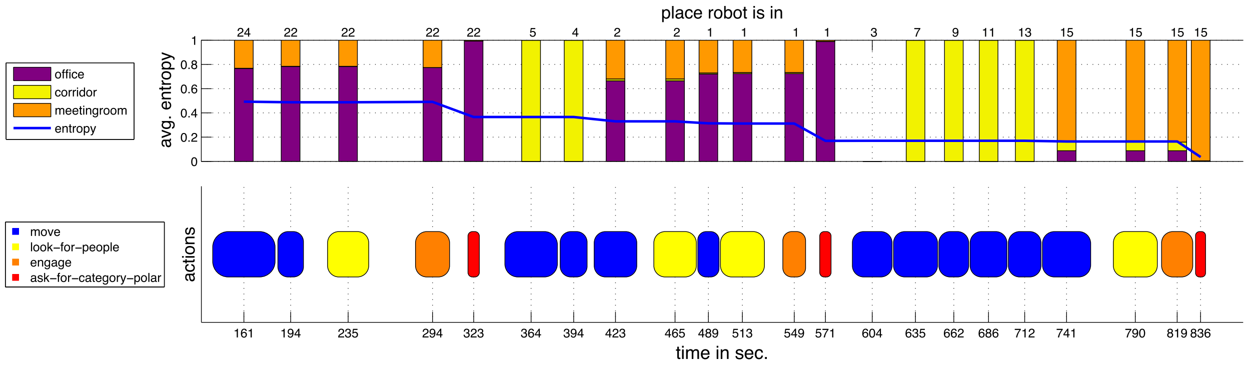


Fig. 18. Course of action and the robot's belief about the categories of rooms in an example categorization run, initiated at the end of the exploration run in Fig. 17 that resulted in the map in Fig. 16(a). Note the monotonic decrease in the robot's uncertainty.

Fig. 17.[25] This figure shows that the average entropy over the room categories declines, but ends fairly high. The robot is particularly uncertain about the categories of Office 1, Office 2 and Meeting room. All seven exploration runs resulted in a fully explored map, taking 1192 ± 156 seconds.[26] Under test condition C_{laser}, the average entropy after exploration ($\hat{H}_{laser} = 0.42 \pm 0.17$) was significantly (with $p = 0.004$ in a t-test) higher than under test condition $C_{combined}$ ($\hat{H}_{combined} = 0.17 \pm 0.15$). This confirms that vision improves categorization.[27] In experiment 2, room categorization is used as the goal to expose how the system plans active information gathering.

7.3. *Experiment 2—categorize: planning with uncertain state and multiple information sources*

Immediately after it completed each exploration run, the robot was given a goal to determine the category of all the rooms in the environment it had previously explored:

```
(forall (?r - room) (K (category ?r)))
```

that is, to know the category of all room instances with high certainty ($\Pr > 0.95$). Two different test conditions, C_{laser} and $C_{combined}$, were again employed. The hypothesis is that by using visual appearance the task will be completed faster. If only the laser is used, the robot will be less certain, and so will choose dialogue actions to determine the categories of the rooms. An example categorization run under C_{laser} is depicted in Fig. 18.[28] This shows that Dora drives to each room, searches for a person (see Section 6) and then asks that person about the room category using a polar question.[29] A typical dialogue follows:

25 In all plots we have collapsed both move and move_direct actions into move.

26 All confidence intervals given in this paper are computed with $p = 0.95$.

27 The system performs categorization in a new environment, having been trained on other environments [36].

28 It is a continuation of Fig. 17, starting with the same uncertainty.

29 Absence of a person, or the person not knowing the answer, will lead to task failure. A lie will cause the robot to draw the wrong conclusion.

Action	Cost	Probability
contains-person-room0-true	0	0.90
category-room2-meetingroom	0	0.83
category-room0-office	0	0.52
category-room3-office	0	0.84
contains-person-room3-true	0	0.90
contains-person-room2-true	0	0.90

Action	Cost	Probability
move dora place24 place 23	2	1.00
move dora place23 place 22	2	1.00
look-for-people dora person1 place22	10	1.00
engage dora person1	1	1.00
ask-for-category-polar dora person1 room3 office	10	1.00
Total:	**25**	**0.27**

Fig. 19. The first few steps of the initial plan during the categorization run depicted in Fig. 18. Note the sequence of assumptive actions, the probabilities of which correspond to the final room category distributions at the end of the exploration run in Fig. 17. The movement (yellow), sensing (red), and dialogue (red) actions have probability 1.00. The likelihood of plan success is the product of the probabilities; 0.27 is therefore the likelihood of the plan succeeding, according to the current belief state. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

```
robot: hello human
human: hello dora
robot: ok
robot: is this room an office?
human: yes
robot: ok
```

The uncertainty about room category drops significantly after each dialogue (timestamps 323, 571, and 836 in Fig. 18). Fig. 19 shows the plan structure: the assumptive actions cope with the uncertainty in room categories, by supposing a particular set of categories is true. Those assumptions are instance assumptions, and thus each assumption's probability is taken from the relational map. The assumptions are also used to hypothesize people in each room who can be asked about the category of that room. Thus, it can be seen how the assumptive actions are used to reason about uncertain worlds with a classical planner. The dialogue actions are chosen when uncertainty is high. This is typically the case when only the laser is used for room categorization.

Under test condition C_{laser}, the robot was able to categorize all rooms in 600 ± 389 seconds. In all these runs the robot had to ask humans to gain the necessary certainty. Under $C_{combined}$, the robot took only 206 ± 121 seconds and only asked the human about the meeting room, since it was already certain enough about the other rooms.

7.4. Experiment 3—find: planning with incomplete state, uncertain state and unreliable observations

The final and most complex task was to find a magazine placed in the environment:

```
(exists (?o - visualobject) (and  (= (label ?o) magazine)
                                  (K (position ?o))))
```

Informally speaking, this goal is to find a magazine. This object search task is an extension of our own work [13,1]. In this article there are three novel features of the experimental conditions. First, diagnostic knowledge is available for the first time, to provide explanations in the case of failing runs. Second, the work by Hanheide et al. [13] was carried out in a closed world, where the map was known in advance, and only room categories were uncertain, while the work by Aydemir et al. [1] was carried out with a cut-down planning domain, so that the robot performed only object search. Finally, for the first time, we deployed the robot in an environment it had not seen before—that is, the room categorization was not trained in the environment it was deployed in. Hence, the robot deals with greater uncertainty than in previous work. Also, we incorporate, for the first time, the ability to search for humans and to engage in dialogue.

Two different test conditions were used for the search task. In the first test condition, C_c, we put the object to be found (a copy of *AI Magazine*) on a table in its *default* room type: the meeting room.[30] In the second test condition, C_{nc}, the magazine was put in Office 2, an unlikely place for the object (`P(mag|office)=0.0475`). In each run, the robot started in Office 1 with no map.

7.4.1. Example run

An example run of test condition C_c (object in meeting room) is shown Fig. 20. Plans from a similar run are shown in Figs. 3 and 4. These show how the assumptive planning is used from the start of the run to hypothesize the necessary entities to produce a plan. Fig. 16(b) shows the map resulting from this run.

[30] Normally, in our experiments, the default knowledge is derived from a combination of the OMICS dataset and probabilities from Web mining. Here, to make the meeting room a likely location for the presence of magazines, we set the default knowledge manually. This modification of the otherwise automatically quantified default knowledge (see Section 6.4) is necessary, because our visual object recognition competence could only recognize objects with flat surfaces. A suitable object that was common in meeting rooms and not so common in offices or corridors was not available. With a state-of-the-art, more flexible object recognizer, this shortcut can easily be evaded. It does not limit our approach in any way.

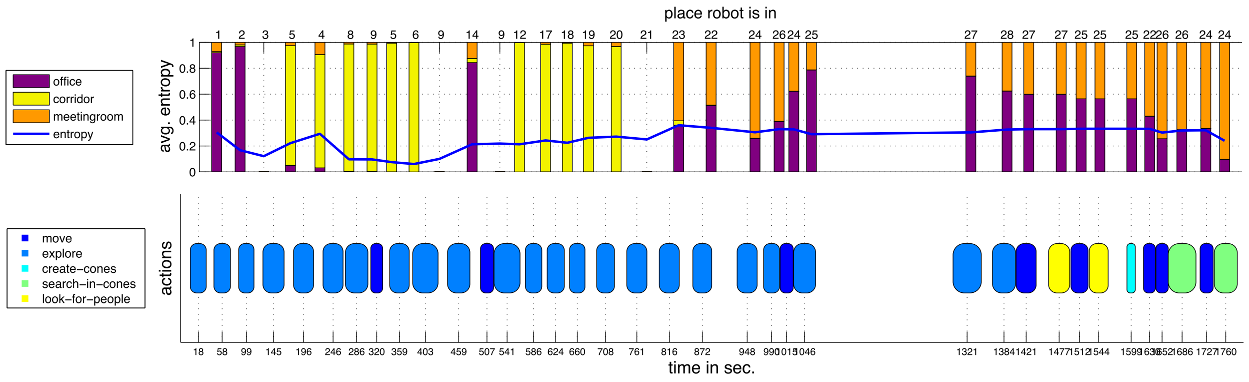


Fig. 20. Course of actions in an example object-search run. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The robot starts in Office 1. To cope with the incomplete map, Dora made essentially the same assumptions described in the running example and shown in Fig. 3 (right). These assume the existence of a meeting room, that an unexplored place-holder leads to that room and that the meeting room contained a magazine. These instance assumptions are based on probabilities from the default-knowledge graph. Thus, if the default probabilities were to change, the assumption probabilities would also change. With this initial plan, Dora explored Office 1, then detected the doorway (timestamp 99 in Fig. 20) to the corridor. Certainty about the categories of both the corridor and Office 1 was obtained quickly via vision. Next, Dora peeked into Office 2 (place 14 at timestamp 459), but correctly recognized it as an office and ignored it, instead exploring the corridor, on the assumption it would lead to a meeting room. Hence, Dora continued to explore until it reached the meeting room (place 23, timestamp 816). Before it could search for the object, the whole room had to be explored. The long waiting time between the explore action at timestamp 1046 and the next action at timestamp 1321 is due to planning. This is caused by the substantial likelihood, in the relational map, of this room being an office (see the top bars in Fig. 20). This makes successful plans less likely—because the magazine is less likely to be in an office—causing planning to be slow.

Owing to the uncertainty in the room categories, the planner then decided that it should first become more certain about the room category, before engaging in visual search. Therefore, Dora searched for a person to ask (at place 27 and place 25, action annotated in yellow in Fig. 20). Since no person could be found, it decided to search for the object anyway (timestamp 1599). The robot generated view cones for this room, and found the magazine on the table when looking from place 24. This example run highlights the ability of the system to plan informed by the degree of belief-state uncertainty. It also shows the use of assumptive actions based on default knowledge to fill in incomplete state information, with their probabilities being generated from the default-knowledge graph. It also shows the switching planner choosing to employ the POMDP planner (green actions in Fig. 20) when faced with unreliable observation actions (visual search). The robot chooses intelligently when to explore, to consult a human, or to start visual search. Dora also successfully replans in the presence of new information (e.g. a new place-holder is discovered) or action failures (e.g. no human could be found in the room).

7.4.2. Quantitative analysis

For the 11 successful runs it took Dora 1340 ± 555 seconds (~ 22 minutes) to find the object. The number of actions executed ranged between 18 and 49 (31.125 ± 10.4). There were a variety of system failure modes. In two runs, the battery simply went flat. In five runs the robot was immobile for five minutes, classified as a timeout, owing to replanning taking too long. This was, in turn, caused by the uncertainty about the room categories being rather high. A milder case of long planning time has been discussed in the example run. This shows that the planner can still fail to quickly return a plan in the face of high uncertainty. The final failure mode was due to errors in the robot's perception system. This includes cases where the robot looked at the object but simply did not see it, cases where the robot had a false-positive detection of an object (run 14) or where it erroneously deleted a place-holder from its map so that exploration stopped (run 9). These cases highlight the opportunity for explanations.

7.5. Experiment 4—explaining failure: planning with diagnostic knowledge

Failures also include cases where the object that the robot searches for cannot be seen, or where it simply does not exist within the search environment. To test the explanation performance of our system, we analyzed the generated explanations for three perception failure runs of the *find* test condition outlined before and an additional seven runs under a specifically designed *explain* test condition, where the object was hidden on a shelf in the meeting room. Of the 10 runs where the search failed, the system was able to generate an explanation for nine runs. Using the optimizations described in Section 5.2, the system generated explanations for runs consisting of up to 50 actions and 22 plans, taking up to 76 seconds to plan. The one run ($e6$) we could not generate any explanation for was of similar size, but contained a number of movement errors in

Table 1
Explanation runs. The first three runs (2, 9, and 14) listed correspond to runs in which the robot did not find the object because of perceptual errors. Runs $e1$–$e7$ are a specific *explain* test condition, in which the object was deliberately hidden (condition C_h). We also report the number of actions the system carried out, the number of explanations generated, the most probable explanation, and the time t it took to generate the explanations. The explanations are detailed on the right.

Run	Condition	No. of actions	No. of explanations	t	Explanations
2	C_{nc}	25	7	0.36 s	e, f
9	C_c	32	6	42.37 s	2 × a, 3 × c, d
14	C_c	6	0	<0.01 s	g
$e1$	C_h	23	2	0.02 s	a, b
$e2$	C_h	14	2	0.04 s	c, d
$e3$	C_h	15	2	0.05 s	c, d
$e4$	C_h	15	3	0.06 s	a, c, d
$e5$	C_h	8	6	0.1 s	f
$e6$	C_h	46	–	–	–
$e7$	C_h	50	6	76.4 s	f

a) A place-holder did not lead to free space.
b) A connection was blocked.
c) A room does not contain a magazine.
d) The magazine does not exist in the environment.
e) A room is a meeting room.
f) Default knowledge: containers are in meeting rooms and the magazine is in the container (six separate assumptions).
g) No explanations were found as the task was deemed successful by the robot owing to false positives in the perception.

addition to the search failure. In this case, the planner failed to find any explanation after exhausting the 2 GB of allocated memory.

7.5.1. *Explanations generated during* find

The system hypothesizes several explanations in each case. These are ranked by likelihood. The most likely explanations are listed in Table 1. The failing cases from the *find* experiment due to perceptual errors are listed as runs 2, 9 and 14 in Table 1. From a post hoc analysis of the robot's representation in these *find* runs we concluded that the actual problem in cases 2 and 9 was a missed place-holder during exploration—that is, the robot tried to reach a place-holder and could not navigate there successfully, causing this place-holder to be deleted from the map and stopping the robot from exploring further. The inference that an initially hypothesized place does not exist can be drawn from such a navigation failure by the explanation system. For instance, in run 9 (condition C_c, object in meeting room), where a place-holder was eradicated owing to a navigation problem, the system could not understand how to reach the meeting room, and so correctly concluded that the object must be outside the known and accessible environment. In run 2 (condition C_{nc}, object in office), however, the system failed to enter Office 2, again owing to an error in the perception of place-holders. As a consequence, the robot found and searched the meeting room (without) the object in it. It concluded that the failure must be due to the object being hidden in a container, a sensible explanation given its high a priori probability of finding magazines in meeting rooms.

7.5.2. *Elicited explanations*

Our system failed only three times under the *find* test condition in a way that permitted generation of an explanation. We therefore conducted another seven search runs, in which the object was hidden so as to ensure failure. Again, the results are as detailed in Table 1. In each case a variety of explanations is posed. Run $e4$ is similar to run 9, with the conclusion that there is no room that contains a magazine. In $e2$ and $e3$ the magazine is hypothesized not to exist. In runs $e5$ and $e7$ the explanation is that there must be some kind of container in the meeting room occluding the object. The preferred explanation depends on the certainty with which the system assumes that it has searched a meeting room. If it is certain that there is a meeting room in the environment, the conclusion drawn is that the object exists, but is hidden; if it is less certain, the likelier explanation is that the object does not exist. In run $e2$, for example, there were no unforeseen movement problems, so the only failure to be explained was not finding the object. As the robot was certain about one room being an office and not so certain about there being a meeting room, it could make the assumption that no magazine was present with high probability (0.82). This, in turn, enabled the assumption that the magazine instance (here the virtual object `object2`) does not exist at all, which is sufficient to explain why the sensing actions did not have the desired result. The generated explanations were as follows:

`object-not-in-room magazine room0 office` $\rho = 0.82$

`virtual-object-nonexistent object2 magazine` $\rho = 1.0$

This is interpreted as "The object does not exist in any room", or put simply, "The object does not exist".

In run $e5$, the most probable explanation was that the object is hidden in some other container. To elicit this case, we put the robot in the meeting room and closed the door to have it search only that room. Also, a human told the robot that it was in a meeting room so that it was absolutely certain about the room category. In this case, the likeliest explanation indeed looks different:

`new-dk-inroom container meetingroom`	$\rho = 1.0$
`new-dk-inobject magazine container meetingroom`	$\rho = 0.8$
`object-in-room-new container room0 meetingroom`	$\rho = 0.9$
`virtual-object-position object0 container room0`	$\rho = 1.0$
`object-in-object-new magazine container object0 room0 meetingroom`	$\rho = 0.9$
`virtual-object-position object2 magazine in object0`	$\rho = 1.0$

Dora here assumes that the object does exist, but it just cannot be seen, because it is presumably hidden in a container. While the explanation from the previous run would be valid for this run as well, its probability would be much lower, as the `object-not-in-room` assumption is much less likely for magazines in meeting rooms than in offices. Dora knows the concept that objects can be inside other objects, as well as that there are objects of type “container”. It has, however, no default knowledge for these in the way it has for magazines in meeting rooms. But we allow Dora to make assumptions about default knowledge: the first two assumptions establish the possibility that containers can often be found in meeting rooms and that magazines can often be found in those containers (the latter has a lower probability of 0.8 on the basis of the existing knowledge about magazines in meeting rooms). The second of those assumptions has the same probability as the assumption that the magazine is in the room. The next two assumptions establish that there is an instance of “container” in the room. The last two assumptions build on the first two, so as to assume that a magazine is in the container. These are the same types of assumptions that are used for task-based planning. Here, we present only the likeliest explanation, but the system can generate an ordered list of alternative explanations.

7.6. *Experiment 5—verifying explanations: planning with incomplete state again*

After creating an explanation, Dora autonomously creates a new goal, via the mechanisms of the goal management system [14], so as to verify the hypotheses made. For the example run above, the goal is to verify the existence of the two assumed objects `object0` (of type container) and `object2` (of type magazine). We ran two additional tests to show the ability of the robot to verify explanations. These were runs starting with a map created by exploration and categorization, but without any idea of object location. In both cases the door to Office 2 was shut. The robot was tasked with finding the magazine, even though it was not within view. On declaring task failure, the robot automatically planned an explanation for the failure, planned a verification of this explanation and executed the verification plan.

Both runs successfully completed all stages up to the execution of the verification plan. The verification plan for one of the two runs can be seen in Fig. 5. This again uses assumptions, but now to hypothesize the existence of a person who can answer the questions needed to verify the robot’s hypothesized explanation of the failure. Thus, it can be seen how assumptive planning is again used to plan in an incomplete world. In these runs, the robot tried to verify both the hypotheses about new instance knowledge and the hypotheses about new default knowledge. To verify the instance hypotheses, Dora approaches the human (once detected) and asks two questions:

```
Robot: Is there a container in the meeting room?
Human: Yes.
Robot: Is there a magazine in the container?
Human: Yes.
```

This satisfies the two instance-knowledge goals as now Dora knows where the object is, even if it cannot be seen. The robot then attempts to verify the default-knowledge hypotheses with two further questions:

```
Robot: Are magazines typically in containers?
Human: Yes.
Robot: Are containers typically in meeting rooms?
Human: Yes.
```

At this point the verification plan has been successfully completed, and Dora can update both the relational map and the default-knowledge graph. In the verification runs, one run completed all four questions, and one run terminated after asking the second question. Nevertheless, the second run completed all the stages of search, explanation and verification, and most of the execution of the verification plan. These complex runs show the ability of the robot implementation of the theory to solve every problem, (i)–(iv), in a single run.

8. Discussion of related work

Our architecture is informed by classic three-layer architectures, such as those proposed originally by Simmons et al. [43] or Peter Bonasso et al. [32], featuring a combination of deliberative and reactive control. This paper has focused on the belief and deliberative layers of our architecture. There are few integrated robot systems that combine domain-independent planning with domain-specific knowledge. Within that community, the need for a high-level *continual planning* and execution monitoring subsystem is widely recognized [52,49,25,56]. The domain-specific knowledge in our approach is structured in three layers, as discussed in Section 1. Various approaches to organizing knowledge in different layers have been developed. Suh et al. [48] present one of the most complete approaches to represent ontological default and instance knowledge in a robot. Also, the KNOWROB architecture of Tenorth and Beetz [50] combines formal, encyclopaedic knowledge with observations from several perception modules, corresponding to our instance and default knowledge. Their representation is one of the few that also supports reasoning about knowledge-gathering actions, but does not support the level of goal-driven planning our system supports and lacks any diagnostic knowledge. Jain et al. [20] present a knowledge representation scheme focusing on learned probabilistic relations. They proposed the use of Markov logic networks and Bayesian logic networks to represent the uncertain default knowledge, an alternative to our use of chain graphs as the underlying representation.

Commonsense, or default knowledge, has been used to guide task planning in robots, such as in the work of Galindo et al. [8], in which a semantic map containing information about typical room-object relationships is used to plan to fill in incomplete information about room types, or the locations of objects. The main contribution of that work is to interleave planning with deductive reasoning about type relations that adds conclusions as they can be drawn, and which reduces the size of the planning domain by pruning irrelevant items. The restrictions on that work are that it assumes a prebuilt metric map, segmented into rooms, and does not reason decision-theoretically either about the value of the information, or the sensor unreliability. Hawes et al. [15,52] also interleave planning with reasoning about a semantic map, but plan with models of the epistemic effects of actions. The use of a continual replanner allows the use of a closed-world planning domain in what is an open world. Whenever new objects or rooms appear, replanning in the new closed world occurs. But unlike the current paper, neither of these approaches explicitly reason about the fact that sensing actions can increase the size of the planning domain, nor the benefits of this for goal achievement. Epistemic effects of sensing actions have also been tackled with a conditional-planning approach that uses the PKS planner [33], which has been applied to planning robot manipulation actions interleaved with sensing actions [25]. This approach models knowledge gained via sensing in a separate database, and also employs a replanning approach to cope with action failure and surprising information.

Probabilistic approaches which do reason about sensing effects have been applied to object search and robot planning in a variety of settings. Velez et al. [51] planned trajectories in a continuous space to maximize the reliability of object detection using a learned observation model. The key contribution is the use of a model of the correlations in sensor behaviour at nearby locations, thus driving the robot to gather more informative views. Martinez-Cantin et al. [29] give a POMDP formulation of active visual mapping, use direct policy search to find a solution, and use Monte Carlo simulation to generate imaginary observations and action outcomes during optimization. The main challenge of decision-theoretic planning in partially observable environments is intractability. Kaplow et al. [22] employed a variable resolution map to achieve scaling with a robotic wheelchair. All three of these approaches concerned path planning in a continuous space. Task-level robot control with a decision-theoretic framework was first tackled by Pineau et al. [34] using a POMDP planner to derive a high-level controller for a mobile robot with a dialogue system by exploiting hierarchy to reduce the state space.

Approaches that mix commonsense and probabilistic knowledge have been developed recently. Zhang et al. [56] combined decision-theoretic planning at the low level, with high-level inference based on non-monotonic logics, although the implementation is restricted to a closed world. Kunze et al. [26] mixed commonsense and probabilistic knowledge in a semantic map of room-object relations in a large-scale space. This is used to support a decision-theoretic planner, which optimizes the object search to trade off between the likelihood of a particular object being present in a location and the cost of visiting that location. The main restriction is that they do not model sensor unreliability, so the decision-theoretic problem is equivalent in complexity to MDP solving. Also, in their work they have a closed world, so their robot cannot explore unknown environments. Hanheide et al. [13] extended this broad approach to partially observable environments. This achieved the scaling required, by employing a planner that switches between decision-theoretic and non-decision-theoretic planning on the fly [11], and showed its application to active visual search. Later, Aydemir et al. [2] showed the first use of assumptive planning to extend this to object search in an open world. Hanheide et al. [13] and Aydemir et al. [2] also provided an advance by employing a semantic map that allows belief updating using different knowledge sources. There are two advances made here that go beyond that work. The first is our extension of the assumptive actions to handle multiple tasks for the first time, such as mapping, categorization or object search. The second is our extension of assumptive planning to generate, and verify, explanations for task failure.

Open-world planning has also been tackled by Talamadupula et al. [49], for a team of robots using the notion of open-world quantified goals, which specify rewards over open sets of objects. The planning approach is a replanning approach similar to the continual planning approach here, and objects are generated as effects of the sensing actions. The approach does not allow the use of assumptive actions or decision-theoretic reasoning. In terms of explaining why a goal cannot be reached, there is some overlap with counterfactual reasoning [27], belief revision [9] and consistency-based diagnosis [39]. All these frameworks deal with identifying a set of propositions or beliefs that either lead to inconsistencies or permit the robot to deduce an observation, while we have a set of operators that allow us to transform states. The idea of detecting discrepancies between expected and observed state also comes up in the goal-driven autonomy framework of Klenk et al. [23], where discrepancies trigger the generation of explanations, which in turn lead to the creation of new goals. There is also a relationship to work in explanation-based learning [30]. In this work, misclassifications led to explanations in terms of a background domain theory that were then used to drive learning.

Explanation for robots has also been tackled with use of event calculus [41,42] where it is posed as abduction of possible causes for perceptual events, within a background theory. Finally, Sohrabi et al. [47] tackled the problem of planning to provide explanations using an extension of linear temporal logic that has the ability to reference past events and action occurrences. An explanation is a set of clauses in the logic, representing an assumption about the initial state, and an executable plan that would cause the observations from that state. Shorter explanations are preferred by using a cost function that sums the formulae required, each weighted by explanation preference rules that are domain specific. By contrast, we employ a probabilistic formulation that prefers more likely explanations.

9. Conclusion

9.1. Summary of contributions

This paper presented a theory of how a robot can plan to gather information and explain failure in environments that have uncertain sensing, uncertain actions, incomplete information and epistemic goals. The abstract version of the theory is concerned with the division of knowledge into three types: instance, default, and diagnostic. In this three-layer division, knowledge at higher layers is used to modify knowledge at lower layers. The instantiation of the theory presented here relies on several technical innovations. We summarize these now.

9.1.1. Joint representation of default and instance state

The combination of instance and default knowledge allows strong inferences to be drawn from relatively little sensory information. The joint belief state forms the initial planning state in each planning session, including determination of the success probabilities of several types of action. The conversion of many different types of knowledge into a single representation allows the use of standard inference methods.

9.1.2. Planning with assumptions

We have described how assumptive planning provides a single mechanism that can be used to solve four problems. First, that of planning in the face of uncertainty. Second, that of planning in open worlds. Third, explaining task failure by hypothesizing both new instance knowledge and new default knowledge. Fourth, verifying that explanation, by hypothesizing the entities needed to verify the explanations. This paper has argued that assumptions are particularly useful when combined with default knowledge. We see assumptions as an elegant way to solve a significant number of problems in robot task planning.

9.1.3. Instantiating assumptions with probabilities

One important requirement for making assumptions is that the robot should avoid making unlikely assumptions. Assigning probabilities to assumptions helps with this, since it gives a principled way to rank sequences of assumptions. Our solution is for assumption likelihoods to be determined by existing knowledge: either instance knowledge or default knowledge. This means that assumptions are not conjured from thin air. It also means that instance assumptions vary according to the changing relational map.

The main contribution of the work is the overall framework for knowledge layering, planning, and state estimation. It is the way that the parts of our approach fit together that is as important as any one part. In the light of this, it is worth reflecting on the open questions that the framework raises, and should be extended to address.

9.2. Questions for future work

The division into three layers of knowledge is predicated on the notion that, normally, when activities are planned, it is most efficient to utilize default knowledge since this reduces the space of plans and outcomes significantly. Planning with explicit representations of all possible default violations generates unnecessary work. It is better to fold in diagnostic knowledge as necessary to explain failure. This is the motivation behind the use of a diagnostic layer. In recent years, the planning community has developed much more efficient algorithms, but even these have limits. To progress, further ways of automatically determining the relevant information for a planning problem are necessary. This can be seen as a problem of determining what subset of instance and default knowledge should be in the attentional frame: the foreground used to create the planning domain. Work on this will be critical to scaling robots to plan with large knowledge bases.

A second problem arises from the attempt to understand the space of mechanisms by which default knowledge can be extended in a principled manner. Our approach to explanations, hypothesizing new default knowledge, is one possible starting point. The larger problem, however, is concerned not just with the mechanisms for hypothesizing new default knowledge, but with automatically searching for new ways of organizing existing default knowledge. Projects such as CYC have been concerned with this, but it is not entirely clear how to exploit their insights in robotics. A third problem is that, in our system, we draw the distinction between default and diagnostic knowledge by hand. In general, this should be done autonomously by the robot. In such a way, the robot could control what it considers as default knowledge, and thus control the efficiency of its regular task planning. Finally, many of the failures robots suffer are because of violations of tacit design assumptions. In the face of this, it is not reasonable to expect an explanation-based approach to fare well, since designers will not readily encode these tacit assumptions in the diagnostic knowledge. But are there ways in which a robot might analyze its execution traces to identify the possible points at which such violations occurred?

Each of these questions is a hard open research problem in its own right. This paper has posed problems of planning under uncertainty, in open worlds, and provision of explanations, in a single framework. Such an overarching framework is also needed for these additional problems, outlined above, and for the many others that remain.

Acknowledgements

We gratefully acknowledge support for this work from the European Community's Seventh Framework Programme, under grant agreements No. 215181, CogX, and No. 600623, STRANDS. We would also like to extend our sincerest thanks to the three anonymous reviewers, and to the editors of the special issue. Their insights and comments hugely improved our paper.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.artint.2015.08.008.

References

[1] A. Aydemir, M. Göbelbecker, A. Pronobis, K. Sjöö, P. Jensfelt, Plan-based object search and exploration using semantic spatial knowledge in the real world, in: Proceedings of the European Conference on Mobile Robotics, ECMR'11, Örebro, Sweden, Sep. 2011, pp. 13–18.
[2] A. Aydemir, A. Pronobis, M. Gobelbecker, P. Jensfelt, Active visual object search in unknown environments using uncertain semantics, IEEE Trans. Robot. 29 (4) (2013) 986–1002.
[3] A. Aydemir, K. Sjöö, J. Folkesson, P. Jensfelt, Search in the real world: active visual object search based on spatial relations, in: Proceedings of the 2011 IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, May 2011, pp. 2818–2824.
[4] C. Bäckström, B. Nebel, Complexity results for SAS$^+$ planning, Comput. Intell. 11 (4) (1995) 625–655.
[5] M. Brenner, B. Nebel, Continual planning and acting in dynamic multiagent environments, Auton. Agents Multi-Agent Syst. 19 (2009) 297–331, http://dx.doi.org/10.1007/s10458-009-9081-1.
[6] H.I. Christensen, G.-J.M. Kruijff, J.L. Wyatt (Eds.), Cognitive Systems, Cognitive Systems Monographs, vol. 8, Springer, Berlin, 2010.
[7] S. Ekvall, D. Kragic, P. Jensfelt, Object detection and mapping for service robot tasks, Robotica 25 (2) (2007) 175–187.
[8] C. Galindo, J.-A. Fernández-Madrigal, J. González, A. Saffiotti, Robot task planning using semantic maps, Robot. Auton. Syst. 56 (11) (2008) 955–966.
[9] P. Gardenfors, Belief revisions and the Ramsey test for conditionals, Philos. Rev. 95 (1986) 81–93.
[10] A.E. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, Deterministic planning in the Fifth International Planning Competition: PDDL3 and experimental evaluation of the planners, Artif. Intell. 173 (5–6) (2009) 619–668.
[11] M. Göbelbecker, C. Gretton, R. Dearden, A switching planner for combined task and observation planning, in: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, USA, 2011, pp. 964–970.
[12] H. González-Banos, J. Latombe, A randomized art-gallery algorithm for sensor placement, in: Proceedings of the Seventeenth Annual Symposium on Computational Geometry, SCG '01, ACM, New York, USA, 2001, pp. 232–240.
[13] M. Hanheide, C. Gretton, R.W. Dearden, N.A. Hawes, J.L. Wyatt, A. Pronobis, A. Aydemir, M. Göbelbecker, H. Zender, Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviours, in: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI-11, Barcelona, Spain, 2011, pp. 2442–2449.
[14] M. Hanheide, N. Hawes, J.L. Wyatt, M. Göbelbecker, M. Brenner, K. Sjöö, A. Aydemir, P. Jensfelt, H. Zender, G.-J.M. Kruijff, A framework for goal generation and management, in: Proceedings of the AAAI Workshop on Goal-Directed Autonomy, WS-4 AAAI 2010, Atlanta, Georgia, 2010.
[15] N. Hawes, M. Hanheide, J. Hargreaves, B. Page, H. Zender, P. Jensfelt, Home alone: autonomous extension and correction of spatial representations, in: Proceedings of the 2011 IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 2011, pp. 3907–3914.
[16] N. Hawes, J.L. Wyatt, Engineering intelligent information-processing systems with CAST, Adv. Eng. Inform. 24 (1) (2010) 27–39.
[17] N. Hawes, J.L. Wyatt, M. Sridharan, H. Jacobsson, R. Dearden, A. Sloman, G.-J. Kruijff, Architecture and representations, in: Cognitive Systems, Springer, Berlin, 2010, pp. 51–93.
[18] M. Helmert, The fast downward planning system, J. Artif. Intell. Res. 26 (2006) 191–246.
[19] H. Jacobsson, N. Hawes, G.-J. Kruijff, J. Wyatt, Crossmodal content binding in information-processing architectures, in: Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction, HRI '08, Amsterdam, Netherlands, 2008, pp. 81–88.
[20] D. Jain, L. Mosenlechner, M. Beetz, Equipping robot control programs with first-order probabilistic reasoning capabilities, in: Proceedings of 2009 IEEE International Conference on Robotics and Automation, ICRA 2009, IEEE, May 2009, pp. 3626–3631.
[21] M. Janíček, Abductive reasoning for continual dialogue understanding, in: M. Slavkovik, D. Lassiter (Eds.), New Directions in Logic, Language, and Computation, Springer, 2012, pp. 16–31.
[22] R. Kaplow, A. Atrash, J. Pineau, Variable resolution decomposition for robotic navigation under a POMDP framework, in: Proceedings of the 2010 IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, USA, 2010, pp. 369–376.
[23] M. Klenk, M. Molineaux, D.W. Aha, Goal-driven autonomy for responding to unexpected events in strategy simulations, Comput. Intell. 29 (2) (2013) 187–206.
[24] T. Kollar, N. Roy, Utilizing object–object and object–scene context when planning to find things, in: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, ICRA'09, Kobe, Japan, 2009, pp. 4116–4121.
[25] D. Kraft, E. Başeski, M. Popović, A.M. Batog, A. Kjær-Nielsen, N. Krüger, R. Petrick, C. Geib, N. Pugeault, M. Steedman, T. Asfour, R. Dillmann, S. Kalkan, F. Wörgötter, B. Hommel, R. Detry, J. Piater, Exploration and planning in a three-level cognitive architecture, in: International Conference on Cognitive Systems, CogSys 2008, Karlsruhe, Germany, 2008, pp. 71–78.
[26] L. Kunze, M. Beetz, M. Saito, H. Azuma, K. Okada, Searching objects in large-scale indoor environments: a decision-theoretic approach, in: Proceedings of the 2012 IEEE International Conference on Robotics and Automation, ICRA 2012, St. Paul, USA, 2012, pp. 4385–4390.
[27] D. Lewis, Counterfactuals, Harvard University Press, 1973.
[28] R. Lienhart, J. Maydt, An extended set of Haar-like features for rapid object detection, in: Proceedings of the 2002 International Conference on Image Processing, vol. 1, ICIP 2002, Rochester, New York, 2002, pp. 900–903.
[29] R. Martinez-Cantin, N. de Freitas, E. Brochu, J. Castellanos, A. Doucet, A Bayesian exploration–exploitation approach for optimal online sensing and planning with a visually guided mobile robot, Auton. Robots 27 (2) (2009) 93–103.

[30] T.M. Mitchell, R.M. Keller, S.T. Kedar-Cabelli, Explanation-based generalization: a unifying view, Mach. Learn. 1 (1) (1986) 47–80.
[31] J.M. Mooij, libDAI: a free and open source C++ library for discrete approximate inference in graphical models, J. Mach. Learn. Res. 11 (Aug. 2010) 2169–2173.
[32] R. Peter Bonasso, R. James Firby, E. Gat, D. Kortenkamp, D.P. Miller, M.G. Slack, Experiences with an architecture for intelligent, reactive agents, J. Exp. Theor. Artif. Intell. 9 (2–3) (Apr. 1997) 237–256.
[33] R.P.A. Petrick, F. Bacchus, A knowledge-based approach to planning with incomplete information and sensing, in: Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems 2002, Toulouse, France, 2002, pp. 212–222.
[34] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, S. Thrun, Towards robotic assistants in nursing homes: challenges and results, Robot. Auton. Syst. 42 (3) (2003) 271–281.
[35] A. Pronobis, 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, St. Paul, USA, 2012, pp. 3515–3522.
[36] A. Pronobis, O.M. Mozos, B. Caputo, P. Jensfelt, Multi-modal semantic place classification, Int. J. Robot. Res. 29 (2–3) (February 2010) 298–320.
[37] A. Pronobis, K. Sjöö, A. Aydemir, A.N. Bishop, P. Jensfelt, A framework for robust cognitive spatial mapping, in: Proceedings of the 14th International Conference on Advanced Robotics, ICAR'09, Munich, Germany, 2009, pp. 1–8.
[38] A. Pronobis, K. Sjöö, A. Aydemir, A.N. Bishop, P. Jensfelt, Representing spatial knowledge in mobile cognitive systems, in: 11th International Conference on Intelligent Autonomous Systems, IAS-11, Ottawa, Canada, 2010, pp. 133–142.
[39] R. Reiter, A theory of diagnosis from first principles, Artif. Intell. 32 (1) (1987) 57–95.
[40] A. Richtsfeld, T. Mörwald, M. Zillich, M. Vincze, Taking in shape: detection and tracking of basic 3D shapes in a robotics context, in: Proceedings of the 15th Computer Vision Winter Workshop, Nové Hrady, Czech Republic, 2010, pp. 91–98.
[41] M. Shanahan, Prediction is deduction but explanation is abduction, in: Proceedings of the 11th International Joint Conference on Artificial Intelligence, vol. 2, IJCAI'89, Detroit, USA, 1989, pp. 1055–1060.
[42] M. Shanahan, Perception as abduction: turning sensor data into meaningful representation, Cogn. Sci. 29 (1) (2005) 103–134.
[43] R. Simmons, R. Goodwin, K.Z. Haigh, S. Koenig, J. O'Sullivan, A layered architecture for office delivery robots, in: Proceedings of the First International Conference on Autonomous Agents, AGENTS '97, Marina del Rey, USA, 1997, pp. 245–252.
[44] K. Sjöö, A. Aydemir, D. Schlyter, P. Jensfelt, Topological spatial relations for active visual search, Tech. rep. TRITA-CSC-CV 2010:2 CVAP317, Centre for Autonomous Systems, KTH, Stockholm, July 2010.
[45] K. Sjöö, H. Zender, P. Jensfelt, G.-J.M. Kruijff, A. Pronobis, N. Hawes, M. Brenner, The Explorer system, in: H.I. Christensen, G.-J.M. Kruijff, J.L. Wyatt (Eds.), Cognitive Systems, Springer, Berlin, 2010, pp. 395–421.
[46] D. Skočaj, M. Kristan, A. Vrečko, M. Mahnič, M. Janíček, G.-J.M. Kruijff, M. Hanheide, N. Hawes, T. Keller, M. Zillich, K. Zhou, A system for interactive learning in dialogue with a tutor, in: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2011, San Francisco, USA, 2011, pp. 3387–3394.
[47] S. Sohrabi, J.A. Baier, S.A. McIlraith, Preferred explanations: theory and generation via planning, in: Proceedings of the 25th AAAI Conference on Artificial Intelligence, San Francisco, USA, 2011, pp. 261–267.
[48] I.H. Suh, G.H. Lim, W. Hwang, H. Suh, J.-H. Choi, Y.-T. Park, Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence, in: Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007, San Diego, USA, 2007, pp. 429–436.
[49] K. Talamadupula, J. Benton, S. Kambhampati, P. Schermerhorn, M. Scheutz, Planning for human–robot teaming in open worlds, ACM Trans. Intell. Syst. Technol. 1 (December 2010) 14:1–14:24.
[50] M. Tenorth, M. Beetz, KNOWROB—knowledge processing for autonomous personal robots, in: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, St. Louis, USA, 2009, pp. 4261–4266.
[51] J.J. Velez, A.S. Huang, G.A. Hemann, N. Roy, I. Posner, Modelling observation correlations for active exploration and robust object detection, J. Artif. Intell. Res. 44 (2012) 423–453.
[52] J.L. Wyatt, A. Aydemir, M. Brenner, M. Hanheide, N. Hawes, P. Jensfelt, M. Kristan, G.-J.M. Kruijff, P. Lison, A. Pronobis, K. Sjöö, D. Skočaj, A. Vrečko, H. Zender, M. Zillich, Self-understanding and self-extension: a systems and representational approach, IEEE Trans. Auton. Ment. Dev. 2 (4) (December 2010) 282–303.
[53] H.L.S. Younes, M. Littman, PPDDL 1.0: an extension to PDDL for expressing planning domains with probabilistic effects, Tech. rep. CMU-CS-04-167, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2004.
[54] H. Zender, G.-J.M. Kruijff, 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, USA, 2009, pp. 1604–1609.
[55] H. Zender, O.M. Mozos, P. Jensfelt, G.-J.M. Kruijff, W. Burgard, Conceptual spatial representations for indoor mobile robots, Robot. Auton. Syst. 56 (6) (2008) 493–502.
[56] S. Zhang, M. Sridharan, J. Wyatt, Integrating probabilistic graphical models and non-monotonic logical inference for robots, IEEE Trans. Robot. 31 (3) (2015) 699–713.