Reasoning and Learning in Real-World Systems for Long-Term Autonomy (LTA)
Papers from the AAAI 2018 Fall Symposium

Using hierarchical expectations grounded in perception for reasoning about failures during task execution

Priyam Parashar€, Ashok K. Goel£ and Henrik I. Christensen€
€ Contextual Robotics Institute, CSE Department, UC San Diego, 9500 Gilman Drive, La Jolla, CA USA
£ Design & Intelligence Laboratory, College of Computing Georgia Institute of Technology, Atlanta, Georgia USA
[pparasha, hichristensen] @eng.ucsd.edu, ashok.goel@cc.gatech.edu

Abstract

Traditionally, meta-reasoning architectures for planning have used abstract representations of expectations about the states of the world. However, embodiment of meta-reasoning on a robot requires grounding the expectations in perception. We propose a dual encoding of expectations based in the concept of occupancy grids. We illustrate this encoding for the task of designing shapes by block placement on a tabletop.

Introduction

Agent architectures for meta-reasoning typically contain three levels or types of information processing: an object level that perceives the world and acts on it; a deliberative level that makes sense of observations of the world and plans actions on world, and a meta-level that monitors and controls the deliberative level through goals and strategies, failures and repairs, and learning and adaptation. Figure 1 outlines a basic general meta-reasoning architecture. (Cox 2005) provides a review of many meta-reasoning architectures; Cox & Raja (2011) provide a more recent anthology of projects on meta-reasoning.

In a meta-reasoning architecture, when an agent reasons about failures, it first generates expectations about the state of the world, then compares the observed state of the world with the expected state, next maps the discrepancy between expected and observed state (the failure) into one or more repairs at the deliberative level (Stroulia & Goel 1995; Murdock & Goel 2008). The recognition of a failure through a comparison of the expected state and the observed state can be challenging if the observations are made through low-level sensors and the expectations are encoded in terms of abstract knowledge representations. Meta-reasoning architectures sometimes use specialized procedures to address this problem (Stroulia & Goel 1999; Jones & Goel 2012). For example, the Augur system uses

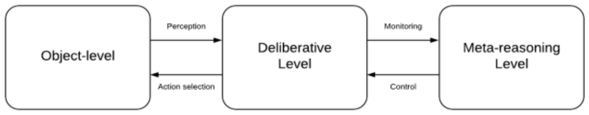


Figure 1. General Meta-reasoning Architecture

specially designed "empirical verification procedures" to map abstract knowledge representations with perceptual observations (Jones & Goel 2012).

For our work on robots capable of meta-reasoning (e.g., Parashar, Sheneman & Goel 2017), we seek a more general strategy for comparing expected and observed states and recognizing failures. In robotics, *seeing* sometimes garners more information than translating it in terms of the presence or absence of pre-defined symbols. Grounding the expectations into perception may have more transparency between the way a robot and the algorithm operating in its mind experience the world. Thus, we propose the use of *dual encodings* of expectations that include low-level visual encodings along with abstract representations.

In particular, in this article, we tackle the problem of designing shapes on a tabletop via block placements. Such a task design problem requires visual details beyond symbolic bindings to be actuated. Our approach uses the concept of *functional indexing in meta-reasoning* (Stroulia and Goel 1995) to encode domain knowledge from visual percepts into a *grid-map* which describes the form of physical placement of objects on the workspace. We propose a hierarchical representation for expectations where top-levels are defined by variables bindings and graph-based block-level symbol relationships, which can then be traced down into the hierarchy to find the grid-map encoding of expected block placement.

Our experimental setup uses the robot Baxter which is furnished with a library of hand-coded hierarchical task networks and annotated expectation databases to plan for drawing shapes. We create example failure cases to compare the meta-reasoning cycle which uses our expectations against one which only uses the symbolic-level.

The rest of the paper gives a quick introduction of relevant concepts and related literature. Next we explain the experimental setup and our methodology for extracting expectation annotations from an image feed. Finally, in the results section we present a qualitative assessment of our system for failure recovery where failures are induced by changing the environmental conditions to mismatch plan pre-conditions at different depths.

Background

We use Hierarchical Task Networks (Erol, Hendler and Nau 1994) to represent the plans for drawing shapes out of blocks. HTNs have two kinds of tasks: primitive actions or nonprimitive tasks. Each goal task is recursively decomposed into sub-tasks until a network of primitive actions remains, which is the base unit of execution, directly triggering some physical actuation in the agent. Nonprimitive tasks on the other hand are compound entities which can be further decomposed into sub-tasks. Each task has a set of pre-conditions and effects attached to it. Pre-conditions define in which state can a task be used to decompose a parent goal and effects depict how the task changes the state. In order to decompose a task into sub-tasks HTN uses the following rule: $\{t\} \rightarrow \{t', t''\}$, if $\text{pre}(t') = \text{pre}(t)$, $\text{pre}(t'') = \text{eff}(t')$ and $\text{eff}(t'') = \text{eff}(t)$. In our system, each task is a hand-coded network which can be executed for drawing the symbol, for example, task **A** is a network which when executed forms the shape A on the table using blocks.

We know from the introduction section above that a meta-reasoner (figure 1) uses expectations to monitor agent's processes, records an execution trace and then reasons over it to find explanations for a failure. A complete meta-reasoning cycle would: (1) note ambiguities using expectations, (2) assess the reason for the unexpected scenario, and (3) reason over the plan trace and environment to guide a solution. Ambiguities are defined as mismatch between expected state description and encountered state during execution, defined by a *mismatch vector* which is a zero-vector of the same size as an expectation's symbolic description with mismatched variables assigned a 1. Finding explanations for ambiguities is a research area of its own and we are focusing on a very narrow range, i.e., unseen configurations of known objects. The current paper only addresses the first two stages of this pipeline for the outlined problem.

Related Work

Another framework relevant to meta-reasoning is that of goal-driven agents (Muñoz-Avila, et al. 2010) which uses expectations to monitor failures, and uses failures as an opportunity to learn new intermediate goals which can help with the overall task goal. Our work has some similarity with (Dannenhauer and Muñoz-Avila 2015), since they too use HTN plans annotated with expectations to conduct meta-level reasoning over their incomplete plans. However, their expectations are of a conceptual form, abstracted on top of environmental symbols.

(Stroulia and Goel 1995) use an explicit "structure-behavior-function" model in to assign blame to various parts of system's current design in a failure case. (Jones and Goel 2012) present "Empirical Verification Procedures" which ground all high-level concepts and axioms known to the agent in lower-level percepts of a video game. (Parashar, Sheneman and Goel 2017) combine meta-reasoning with reinforcement learning using purely visual form expectations. However, they still use symbolic descriptions or computerized descriptions of visual which simplifies the perception part of the problem.

In robotics, (Beetz, Mösenlechner and Tenorth 2010) and (Cox, et al. 2016) have used reasoning over abstracted concepts to help with low-level manipulation and task planning in robotics. However, all these methods use the symbolic layer to integrate reasoning with the environment.

Approach

Experimental Setup

We are using Mega Bloks™ to draw shape and for simplicity will be referring to a single unit as a block. Our system considers two different shapes of blocks: 1x1 and 1x2; and supports two different colors: blue and red. Goals are communicated as strings naming the shape to be drawn. Each block's physical placement is described by two attributes, its orientation with respect to the table's axis and the location of its centroid in the workspace (figure 1). When blocks are recognized in an image they are indexed with a number starting at 0, e.g. $b^0 = \{\text{color}, \text{ shape}\}$. To describe the placement of two blocks with respect to each other we use a graph-based format where $R_{0,1}$ is a transform which when applied to the orientation and location of centroid of block b^0 would result in the centroid of block b^1. A 1x1 sized Mega Blok is of length 6.1 cm and width 6.1 cm, which we denote as l_b in the rest of the paper.

In order to codify the pre-conditions and effects of the HTN tasks we use a symbolic state descriptions which include: (a) ob_{grip}: a single-value set depicting the block in the gripper, φ if empty and $\{i, j\}$ if a block of color i and shape j is in the gripper, and (b) B: set of pairs depicting the required blocks and their availability. Other variables have been abstracted because they are not relevant to current discussion. Similarly, the only primitive action is: *place(b, x, y)* which subsumes smaller actions within it.

Hierarchical Representation of Expectations

The planning framework is still only using symbolic descriptions since we do not want the high-level planner to be bogged down by the details of the scene, however if a failure is noted we need access to a deeper knowledge-base which is encoded in our expectations. Hierarchical expectations are coded so that the 0th level has block-symbols describing the resultant composition effect an action and on the next or 1st level, we have cropped visual *grid-maps* centered on each block to capture a locally detailed description of its placement. Each grid-map is of length $3l_b$x$3l_b$ to include the block and some of the surrounding context. The plans are annotated with expectations at the primitive action level and backtracked to the goal task-level by assigning the parent task the expectation of last primitive action in its decomposition.

The expectation extractor uses a top-view image feed of workspace, via the robot's eye-in-hand setup, to extract expectations associated with each stage of the hierarchical task network. The block-level description of a symbol is extracted by performing HSV color-thresholding for blob detection on the tabletop view of the symbol under-construction. Once a colored blob is found, its shape is assigned by comparing blob-axis with l_b. Next, the visual expectation is a cropped centered on the centroid of the blob, and a quantized view of the form of the blob, i.e., a

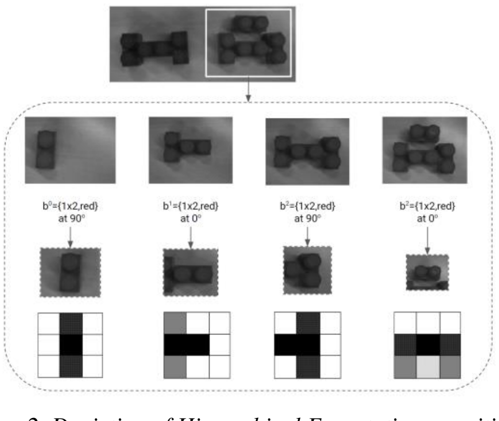


Figure 2. Depiction of Hierarchical Expectations, position of centroid removed from symbolic description for brevity. At the top, darker H shape is blue while lighter A is red.

grid-map is created. A grid-map is like an occupancy grid where the occupancy of a cell is decided based on color presence of the block on a uniformly colored background. The resolution of grid-map is $\frac{1}{2}l_b$.

It is to be noted that such a low-level description would require domain knowledge to be encoded since its form is tightly integrated with the goal of the problem itself. In the current implementation, the HTN plan is executed by an expert kinesthetically driving the agent to annotate the plan with resulting "ideal profile" of expectations. After the full execution, the expectation annotator creates two kinds of databases, one of annotated plans, denoted as P_a, and another one of the low-level grid-maps and symbolic expectations annotated by the causing action(s) which points back to the parent task itself, denoted by E_{db}. If multiple instantiations collide with the same expectation entry, then that expectation is annotated with all the corresponding action labels. The second database is key for a two-way communication between sensor information and plan knowledge even when symbol grounding fails during runtime.

Experimental Results

As one can see by looking at the state description, the failure can be of logical or physical kind. By physical we mean misplacement of gripper, wrong state of gripper, etc. This paper does not address these failures. In the rest of the paper when we explain our algorithm we are addressing only the logical failures, i.e., missing blocks, unexpected configuration of blocks, etc. We broadly classify the failures into two kinds, one where known entities are observed in an unseen configuration thus going against the explicit nature of pre-conditions, and one where unknown entities are observed breaking the planner's assumptions. We present here an example case where we create both kinds of failures by manipulating the environment. In this example, we have provided the agent with the plans for shape *A* and *H* (figure 2), using two 1x1 blocks for H rather than one 1x2 block. We use these shapes because they possess the kind of form similarities we want our algorithm to identify. We want to see if our expectations can help in creating connections between pieces of knowledge already stored in our database better than symbolic expectations. Next, the environment is modified to progressively make the failure more difficult for drawing the shape A. We replace required 1x2 block with another:

- Block of same shape but different color
- Set of two 1x1 blocks of same color
- Set of two 1x1 blocks of different color

The mismatch vector is used to identify the 1st instance of action in the invoked task which uses the mismatched entity, i.e., block in our case. Next, this action's expecta-

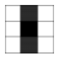
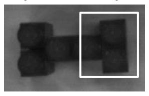
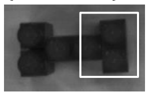
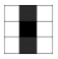
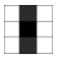
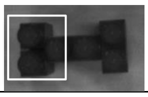
Type of Replacement	Affected Action-Exp Pair	Grid-map Match	Symbolic Match
1x2, red → 1x2, blue	b^0 = {1x2,red} at 90° 	b^4 = {1x2,blue} at 90° 	b^4 = {1x2,blue} at 90° 
1x2, red → 1x1, red + 1x1, red	b^0 = {1x2,red} at 90° 	None	None
1x2, red → 1x1, blue + 1x1, blue	b^0 = {1x2,red} at 90° 	b^2 = {1x1,blue} at 0° 	None

Table 1. Summary of Match results. The white square shows which block's resultant placement expectation in shape H was matched

tion is retrieved from P_a and a nearest-neighbor algorithm is invoked to find ranked matches from E_{db}. We compare the entries retrieved by symbolic matching and grid-map matching to qualitatively assess the usefulness of our hierarchical expectation representation.

Results and Discussion

Our results are summarized in table 1 and compare the grid-map retrievals against symbol expectation matching. The most significant result is shown in row 3 where due to functional encoding of grid-maps its matches were able to search for a visual similarity of form unlike symbolic matching. For row 2, neither found a match since no shape uses {1x1, red} blocks in the current HTN plan library.

Our approach lends itself naturally to hybrid execution architectures where reactive learners manipulate raw-data and work in synchrony with deliberative planners which rely on some heuristic or some other form of domain knowledge. While it is easy to think of meta-reasoner as only an additional layer, its strength lies in enabling trading of valuable information across these two layers. It is this strength of meta-reasoner to form a global view which we believe will be a valuable addition to the long-term autonomy literature in robotics. Specifically, its across-event reasoning can augment the strength of episodic performance exhibited by reactive learners and task-oriented planners.

Conclusion

We presented in this paper dual encoding of expectations used in meta-reasoning to ground abstract representations of world states within perceptual encodings. We performed an experiment by changing our task environment to break pre-conditions coded for the required task and compared the visual expectations against the abstract expectations to see which form can help in recognizing failure as a first step towards failure recovery. Our results match our hypothesis: by encoding functional and visual form of the world within expectations, the meta-reasoner can make better connections within its knowledge base.

References

Beetz, M., L. Mösenlechner & M. Tenorth. 2010. CRAM - A Cognitive Robot Abstract Machine for everyday manipulation in human environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems.* 1012-1017.

Cox, M.. 2005. Field Review: Metacognition in Computation: A Selected Research Review. *Artif. Intell.* 169: 104-141.Cox, M. , Z. Alavi, D. Dannenhauer, V. Eyorokon, H. Muñoz-Avila & D, Perlis. 2016. MIDCA: A Metacognitive, Integrated Dual-Cycle Architecture for Self-Regulated Autonomy. *AAAI.* 3712-3718.

Dannenhauer, D & H. Muñoz-Avila. 2015. *Goal-Driven Autonomy with Semantically-Annotated Hierarchical Cases. Case-Based Reasoning Research and Development*, 88-103.

Erol, K., J. Hendler & D. Nau. 1994. HTN planning: Complexity and expressivity. *AAAI.* 1123-1128.

Jones.,J. & A. Goel. 2012. Perceptually grounded self-diagnosis and self-repair of domain knowledge. *Knowledge-Based Systems* 27: 281-301.

Muñoz-Avila, H., U. Jaidee, D. Aha & E. Carter. 2010. Goal-Driven Autonomy with Case-Based Reasoning. *Case-Based Reasoning. Research and Development,* pp. 228-241.

Parashar, P, B. Sheneman & A. Goel. 2017. *Adaptive Agents in Minecraft: A Hybrid Paradigm for Combining Domain Knowledge with Reinforcement Learning.* In Procs AAMAS-2017, pp. 86-100.

Stroulia, E, & A. Goel. 1995. Functional representation and reasoning for reflective systems. *Applied Artificial Intelligence* 9: 101-124.

Stroulia, E., & Goel, A. 1999. Evaluating Problem-Solving Methods in Evolutionary Design: The Autognostic Experiments. *Human-Computer Studies* 51:825-847, 1999.