Developing Grounded Goals through Instant Replay Learning

Lisa Meeden
Computer Science Department
Swarthmore College
Swarthmore, PA 19081, USA
meeden@cs.swarthmore.edu

Douglas Blank
Computer Science Department
Bryn Mawr College
Bryn Mawr, PA 19010, USA
dblank@cs.brynmawr.edu

***Abstract*—This paper describes and tests a developmental architecture that enables a robot to explore its world, to find and remember interesting states, to associate these states with grounded goal representations, and to generate action sequences so that it can re-visit these states of interest. The model is composed of feed-forward neural networks that learn to make predictions at two levels through a dual mechanism of motor babbling for discovering the interesting goal states and instant replay learning for developing the grounded goal representations. We compare the performance of the model with grounded goal representations versus random goal representations, and find that it is significantly better at re-visiting the goal states when using grounded goal representations.**

I. Introduction

In order for a developmental robot to be able to move about in the world in a purposeful, self-motivated manner, it first needs to learn to understand the moment-to-moment consequences of its actions. From this reactive foundation it can then begin to assemble longer-term plans to take it from its current state to more distant desired states. How could such plans be represented? How could they arise from experience and be grounded in the robot's perceptual framework? To explore these questions we have designed a developmental architecture to enable a robot to explore its world, to find and remember interesting states—those moments when its sensory state changes dramatically—and to associate goals with these interesting states so as to be able to achieve them again in the future.

Ultimately, we want the system to be able to predict more abstract information about itself, such as its own goals. However, it is impossible to anticipate as yet unknown goals. How could the robot know that a goal is just a few steps away so that it could predict it? To solve this chicken and egg problem, we let the robot motor babble, wandering around until it stumbles upon an interesting sensory situation. Once found, we associate a goal with this sensory state. That is, we keep a short history of recent past states, and we replay this history, training the system to predict what it could not have known: that a goal was right around the corner. The goal is an internally generated representation, called a *protoplan*, that reflects the model's own understanding of how it arrived at this interesting sensory state. Through instant replay learning, the robot can eventually develop representations to anticipate future goal situations before it has actually encountered them. Instant replay learning is similar in spirit to *experience replay* [1], which was most recently used to great success in deep reinforcement learning applied to Atari video games [2].

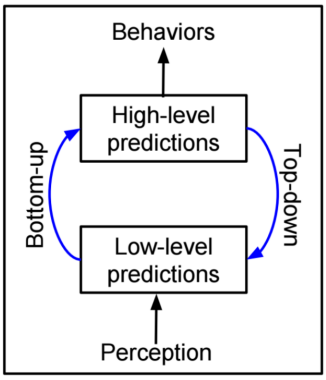

Fig. 1. Hierarchical model. The lower level learns to predict perceptions (including proprioceptions), and as a side effect creates grounded abstractions of its current perceptual state. The higher level learns to predict these grounded abstractions (bottom-up) and can manipulate the lower level by setting its internal state (top-down) to achieve longer-term behaviors.

To allow the robot to build up an understanding of its actions in the environment and to develop its own goals, we use predictions at two levels. Figure 1 provides an overview of this approach. First, at the lower level, the robot learns to predict the effects of its own motor actions on its perceptual state. Next, at the higher level, the robot learns to predict its own abstracted internal states that were developed at the lower level. This bottom-up process starts with concrete sensory-motor experience, and leads to the development of more abstract representations. Finally, given these abstractions, the higher-level can control the lower-level in a top-down manner, by manipulating the lower-level's internal states to produce desired behavior.

This paper describes a hierarchical neural network model that develops its own goals and uses instant replay learning to build an understanding of how to achieve them.

II. Related Work

Our model is based on several fundamental features.[1] First, the model is trained via prediction, allowing it to bootstrap its own learning. Second, the model is hierarchical, with both bottom-up and and top-down interactions. Third, the model develops its own goal representations. Fourth, the model is given an attention mechanism allowing it to focus on key interactions with the environment. Related work associated with each of these features is discussed below.

Many researchers have noted the central importance of prediction in cognition. Elizabeth Spelke has described prediction as part of the core knowledge on which cognition is built [4]. Andy Clark describes prediction as "a form of self-supervised learning, in which the 'correct' response is repeatedly provided, in a kind of ongoing rolling fashion, by the environment itself [5, p. 18]." Jeffrey Elman has noted that prediction is a useful task because its solution requires that the learner be sensitive to the temporal structure of the environment [6].

The power of hierarchical models is now being more widely recognized. Andy Clark argues that "prediction-driven learning operating in hierarchical (multilayer) settings plausibly holds the key to learning about our kind of world: a world that is highly structured, displaying regularity and pattern at many spatial and temporal scales, and populated by a wide variety of interacting and complexly nested distal causes [5, p. 19]." Jun Tani also emphasizes that hierarchical systems, with both top-down and bottom-up processes, are essential, stating that "it is the interaction of these two processes which is seen as central to understanding mind [7, p. 7]."

Tani proposed a hierarchical developmental robotics model based on a new type of recurrent neural network known as a RNNPB, where PB stands for *parametric biases* [8]. These biases are provided as additional inputs to the network and can be used by higher-level layers to exert top-down pressure, serving the role as goals to invoke certain learned behaviors. In his initial experiments, these biases were set by the experimenter, and later tuned by the system. In our model, the system develops its own grounded goals, based on experience.

There are many different approaches to modeling attention in developmental robotics. One possibility is to build into the model an idea of *distinctive states* [9], which are unique landmarks in the world that should be remembered and used to navigate. Another possibility is that the robot identifies situations that it cannot currently predict [10]. However, focusing on prediction error alone may be problematic due to noise. Another option is to monitor prediction learning progress, and focus on situations where learning progress is high [11], [12]. In this experiment, we define a concept of *interest* that is based on changes in moment-to-moment sensor states; situations when a robot's sensor state changes dramatically are worth remembering.

[1]For further examination of fundamental features in developmental robotics, see [3].

III. Experimental Methods

A. Robot and Environment

For these experiments, we used a simple robot simulator, called Jyro [13]. The Jyro simulator is written in the Python programming language and is designed to model actual physical robots, such as the Mobile Robotics Pioneer robot used here. The robot's movement is controlled by two motor commands: translation and rotation. For the experiments described here, the robot was equipped with a total of 19 sensors: 1 stall sensor, 2 light sensors, and 16 sonar sensors. The stall sensor is discrete (either 1 when stalled or 0 when not), while both the light and sonar sensors are continuous (normalized to be in the range $[0, 1]$). The robot was trained in a 4x4 meter walled enclosure with a single light source, as shown in Figure 2. Although the simulator is based on an actual robot and sensors, for these experiments we have not introduced any noise into the robot's movements or sensor readings. However, the simulated robot can only sense its world using local sensor observations and does not have access to its true global state. This leads to perceptual aliasing—the situation where many different locations in the world are indistinguishable from one another based on sensor readings alone.

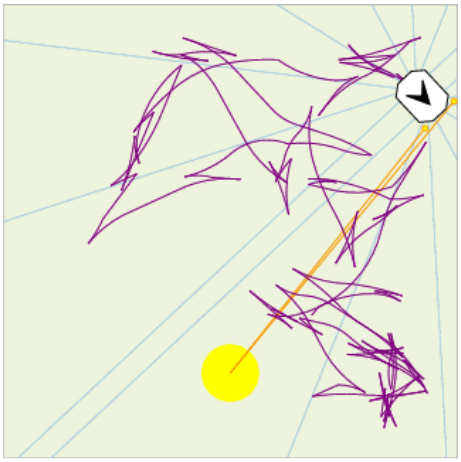

Fig. 2. Visualization of the robot and environment from the Jyro simulator. The light source is shown as a large yellow circle in the bottom center of the world. The robot is shown in white. The black arrow head in the center of the robot depicts its current heading. The two light sensors are depicted as small circles on the front of the robot. The two orange lines emanating from each light sensors indicate that the robot has a line of sight to the light source. The 16 sonar sensors are shown as blue lines emanating from the robot. The purple line indicates the robot's path over 1000 steps of motor babbling.

B. Neural Network Model

Figure 3 shows a detailed view of the hierarchical neural network model shown in Figure 1. The Perception Prediction Network (bottom) is trained to predict the next sensory state and to reproduce the current motor command when given the current sensory state and a motor command. As a side-effect of learning to successfully predict its sensory-motor state,

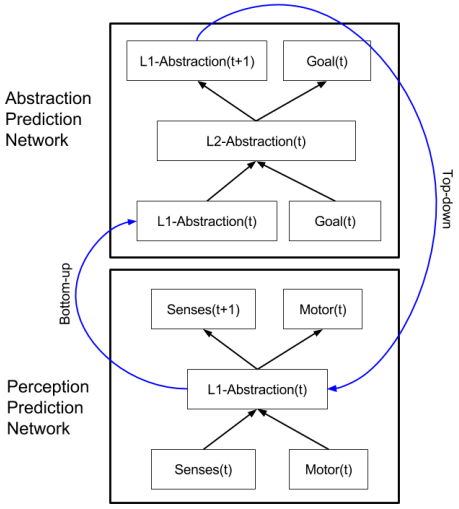

Fig. 3. Neural network model. Straight, black arrowed lines represent fully-connected, feed-forward weights between named groups of units. Curved, blue arrowed lines indicate the ways that the two networks interact with one another. The size of each group of units was, Senses:19, Motor:2, L1-Abstraction:25, Protoplan:25, and L2-Abstraction:50. The bottom network has 1,096 parameters and the top network has 5,100 parameters.

the Perception Prediction Network develops abstractions in its hidden layer (called L1-Abstractions in Figure 3).

The input and hidden layers of this bottom network can be thought of as an *encoder*—when given a sensory-motor state, the encoder produces an encoded abstraction of it. The hidden and output layers of this bottom network can be thought of as a *decoder*—when given an encoded sensory-motor state, the decoder produces the next anticipated sensors and the current motor. These L1-Abstractions contain grounded information based on the robot's current sensory-motor experience. The Abstraction Prediction Network (top) is trained to predict these L1-Abstractions in the context of a particular goal it is trying to achieve.

This hierarchical model can be trained in either an online or offline manner. Online learning is a much more difficult task for the model as the top network must learn the bottom network's representations as it is still adapting, creating a moving target problem. Although online learning is a more difficult task (and takes much more computational time), it may have developmental benefits. For example, it may be beneficial to have the Abstraction Prediction Network affect the lower-level perceptual predictions.

For the experiments reported here, we start with the simpler offline learning. Although offline learning prevents the higher-level network from informing the lower-level network, it allows a straightforward, efficient two-phase learning process. It should be noted that our two-phase offline learning process still contains the essence of online development (explained below in subsection D).

In phase one, the Perception Prediction Network is trained until accuracy peaks (500 epochs), then weights of this network are fixed. In phase two, the Abstraction Prediction Network is trained until accuracy peaks (200 epochs), and the weights of this network are also fixed. Testing is done after both network's have completed training.

The neural network model is implemented in Keras [14] on top of TensorFlow [15]. The model uses the *mse* (mean squared error) loss function with the *adam* optimizer. All units use the *tanh* activation function.

C. Goal Discovery Through Motor Babbling

The data set for training the network model is generated through motor babbling. The robot wanders through its environment randomly exploring its motor space. During motor babbling, a new random motor action is generated every five time steps, allowing the robot to repeat the same random action for a short sequence of time.

As the robot explores the world through motor babbling, there will be instances when its movement causes a large change in its moment to moment sensor readings. The model defines *interest* as the sum of absolute differences between the previous sensors and the current sensors. When interest goes above a given *interestThreshold* (0.475 in these experiments), goal discovery is triggered.

We want the model to recognize interesting state changes of this kind as situations that should be remembered and should be repeatable in the future. To do this we will designate the current sensor state at the time of a spike in interest as the goal's *endState*. We also record the recent precursor states that led up to the discovery of this interesting state. We go back *recallSteps* in time (10 in these experiments), saving the sequence of sensory-motor states the robot experienced to be able to train it to anticipate each goal.

In order to effectively use these goal *endStates*, the model maintains a goal memory. At the beginning of training this goal memory is initially empty. Whenever the interest level rises above the *interestThreshold*, the current state is compared against all of the previously found goal *endStates*. If the Euclidean distance between that current state and any of the existing goal *endStates* is below a *distanceThreshold* (0.19 in these experiments), then it is considered a match for that previous goal. Otherwise a new goal instance is created, and the current sensors are saved as the goal *endState*.

Note that the number of goals discovered during motor babbling is dependent on the settings of the *interestThreshold* and *distanceThreshold*. These thresholds must be tuned appropriately for a particular robot and environment.

Motor babbling was done for 100,000 steps and all applicable sensory-motor data was saved along with information about each goal that was discovered. Twenty unique motor babbling runs were conducted to serve as the basis for training the model.

D. Instant Replay Learning

The point of instant replay learning is to allow the robot to randomly stumble onto interesting sensory experiences, and to then rewind a short bit of memory in order to learn the steps that lead up to that state. In online learning, this

merely requires the model to keep a short, constantly updating buffer of memory (one for the Perception Prediction Network, and one for the Abstraction Prediction Network). In offline learning, the system must keep track of the entire set of these memory sequences so that it can be efficiently trained.

How should a goal be represented in instant replay learning? We desire an abstract, yet perceptually grounded way to trigger the model to execute the sequence of actions that lead to a particular goal *endState*. The activations of the hidden layer of the Perception Prediction Network offer an ideal representation. Thus immediately after an interesting situation has occurred, the contents of the hidden layer will be designated as the goal representation. Meeden called these representations *protoplans* and demonstrated in previous work that they could be used to successfully guide behavior in neural network learning [16].

Our hypothesis is that discovering goals based on experience and building goal representations grounded is this experience will lead to better performance at reproducing the behavior needed to achieve these goals than a system that uses arbitrary goal representations. We conducted an experiment to test this hypothesis. In the next section, we provide a focused set of results for a typical run of one experiment. Then we provide evidence based on a series of 20 experiments that support our hypothesis.

IV. The Anatomy of One Experiment

The model begins each run in motor babbling mode, where the robot explores its environment and discovers interesting states. When the interest level rises above the *interestThreshold*, because two successive sensor states differ significantly, goal discovery is triggered. If the current sensor state is different enough from all existing goals, then a new goal will be created. Figure 4, shows all of the goal *endStates* found in one experiment. Note that many of the goal *endStates* are associated with the stall sensor, which is triggered when the robot bumps into a wall, and are thus clustered around the edges of the environment. There is another cluster of goal *endStates* around the light source where the robot's light sensors pick up larger variations in light readings.

Fig. 4. The *endStates* (depicted as arrow heads) of all 169 goals found in one run. The paths the robot took during the 10 actions before reaching each *endState* are also shown as black lines attached to each arrow head.

Figure 5 shows how the number of goals discovered during the experiment grows over time. During the first 20,000 steps, the rate of goal discovery is quite high. As the robot continues to explore its environment, there are fewer and fewer new interesting situations to find, and new goal discovery slows dramatically. On average, 162.5 goals were found across the twenty motor babbling runs.

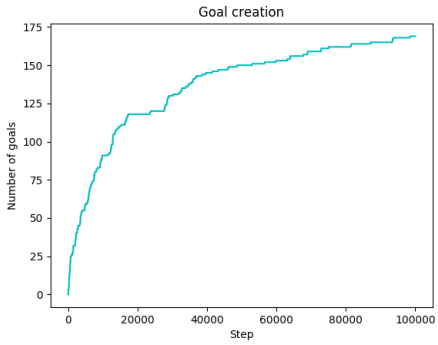

Fig. 5. Goal creation over time.

A. The Anatomy of Two Goals

Let's consider two specific goals found in the run discussed above (shown in Figure 6). Because each goal is created based on a high interest level, we can summarize goals in terms of which of their sensors experienced the most change. For both goals 24 and 110, the sonar sensor positioned front and center on the robot changed the most. Sonar sensors measure distance to obstacles at a particular angle relative to the robot. For goal 24, this sonar value decreased as it approached the south wall. For goal 110, this sonar value increased as it backed away from the north wall.

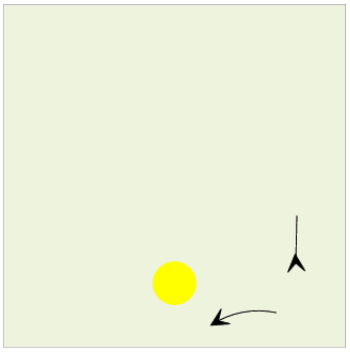

Fig. 6. Goal 24 is at bottom. The robot was making a series of forward arcing movements when the goal was discovered. Goal 110 is on the right. The robot was making a series of straighter backward movements when the goal was discovered.

B. Generative Behavior

In order to analyze how the model has represented the motor sequences associated with each goal, we can clamp on a

particular protoplan and produce a sequence of motor actions and record how the robot behaves as a result. Generating a motor sequence requires both the lower network and the upper network to work in tandem. The lower network decodes the sequence of abstractions produced by the upper network to determine how the robot should move. The entire process begins by encoding the robot's initial sensor state with a no-op motor command. This serves as the first L1-Abstraction that is combined with the protoplan to initiate the sequence in the upper network. In this generative mode, the robot is choosing actions autonomously based on its learned goals.

Figure 7 demonstrates that when running the model in generative mode, we can place the robot in a variety of locations, and by changing only the protoplan input, the model produces the appropriate behavior associated with each goal. For goal 24, the model generates arcing forward motor sequences that reduce the front sonar reading. While for goal 110, the model produces backward motor sequences that increase the front sonar reading. Additionally, Figure 7 demonstrates that the model has generalized how to respond appropriately to each goal in different contexts from which it was trained. Recall that the model was only trained on the short sequences from motor babbling where goal's were initially discovered.

C. Protoplan Relationships

We have looked in detail at two particular goals discovered during one run of the experiment. Let's now place these two goals within the larger context of other goals found throughout the entire run. Figure 8 depicts a cluster analysis done on the protoplans of the top 50 most-matched goals. As our analysis in the previous sections has shown, goals 24 and 110 are quite different from one another, representing opposite situations (increasing vs decreasing front sonar values) and opposite behavior (forward vs backward movement). This is clearly reflected by their distant relationship in the cluster analysis. Yet, because all of these protoplans are grounded in the robot's sensory-motor experience, they are richly interrelated.

V. Overall Experimental Results

Our hypothesis is that grounded goal representations will outperform arbitrary goal representations in reproducing the trained goal behavior. To test this hypothesis we compared the performance of our model with a second instantiation of the model in which the Perception Prediction Network remains unchanged, but the Abstraction Prediction Network is trained with randomly generated goal representations rather than protoplans. To be clear, in the control experiment, each goal is assigned an arbitrary, but unique goal representation. Whenever that goal is being trained, the same arbitrary representation is being used. All other aspects of the experiments remained the same. Twenty different lower-level networks were trained and paired with 20 different protoplan upper-level networks and also paired with 20 different control upper-level networks.

To test the performance of the models, we put them in generative mode, placing the robot at the location 10 steps prior to where each goal was originally discovered. We clamped on the appropriate goal representation—protoplans for the original model and the arbitrary representation for the control model—and propagated each model 10 steps forward. We computed the Euclidean distance between the final sensor state reached and the goal's *endState*. The closer this distance, the better the model has learned to achieve the desired goal. We found that in a paired t-test, the sum of distances generated by the original model were significantly lower than that generated by the control model ($p < 0.01$) indicating that the grounded goal representations were a better foundation for learning.

VI. Conclusions

In this paper we have shown that a robot with a built-in concept of interest combined with instant replay can learn to associate states of interest with protoplans generated internally by its feed-forward network models. These protoplans can be used as goals in order to produce multi-step motor sequences which move the robot into specific sensory situations when applied in appropriate starting conditions. These grounded and autonomously created protoplans elicit differing behaviors in the robot. The protoplans define a continuous space of representations that can further develop over time.

The model described in this paper captures the essence of a developmental system exhibiting emergent properties from a high-level conceptual layer, and a low-level perceptual layer. Yet, there are many possibilities for further exploration. For example, although we have focused on predicting the next steps in sequential movements, the model uses only feed-forward networks. An obvious enhancement would be to use recurrent networks.

We also plan to enhance the concept of interest and to explore other mechanisms for goal discovery. The concept of interest used in these experiment is simplistic. It is only triggered by large instantaneous changes in sensors, but not by more gradual changes over the course of many steps. Also, measuring interest directly on raw sensor values will likely not scale up well to more complex sensors such as cameras where images can have thousands of pixel values. We ultimately want to test the model on real, physical robots.

VII. Acknowledgments

We would like to thank Jim Marshall for his thoughtful suggestions and his willingness to discuss these ideas with us in depth. We would also like to thank the anonymous reviewers who provided constructive feedback that led to a much improved final version of the paper.

References

[1] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, vol. 8, no. 3, pp. 293–321, 1992. [Online]. Available: http://dx.doi.org/10.1023/A:1022628806385

[2] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015.

[3] D. Blank, J. Marshall, and L. Meeden, "A developmental robotics manifesto," *IEEE CIS Newsletter on Cognitive and Developmental Systems*, vol. 14, Spring 2017.

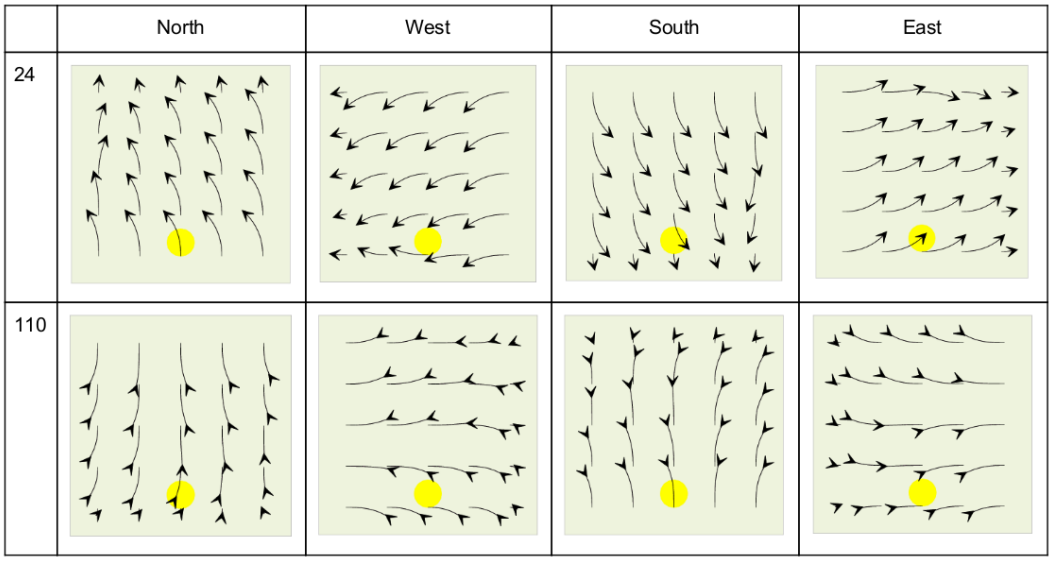

Fig. 7. Comparing the model's generative behavior for goals 24 (top row) and 110 (bottom row). The starting locations are from a 5x5 grid of (x, y) positions using headings north (col 1), west (col 2), south (col 3), and east (col 4). Each trajectory represents 10 steps.

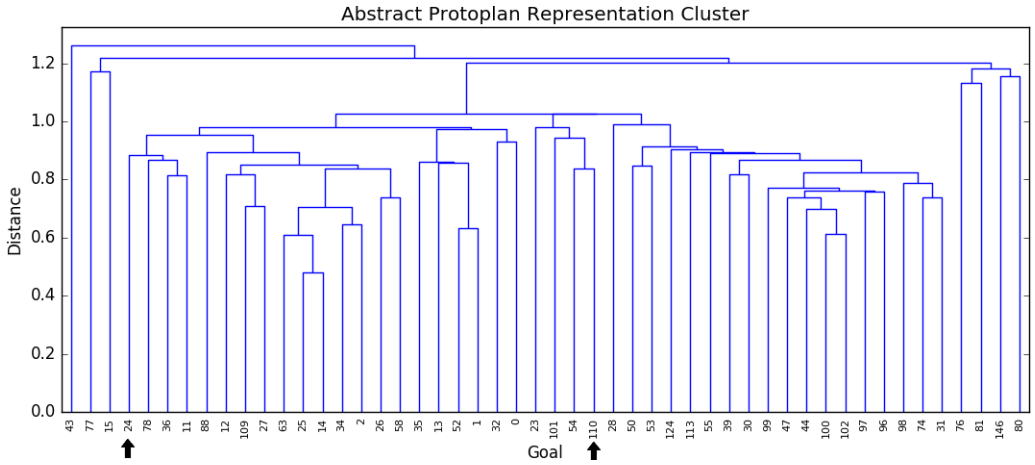

Fig. 8. Cluster of the protoplans for the top 50 most-matched goals. Goals 25 and 45, which were analyzed in depth, are highlighted with arrows.

[4] E. Spelke, "Core knowledge," *American Psychologist*, vol. 55, no. 11, 2000.

[5] A. Clark, *Surfing Uncertainty: Prediction, Action, and the Embodied Mind*. Oxford University Press, 2016.

[6] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, 1990.

[7] J. Tani, *Exploring robotic minds: Actions, symbols, and consciousness as self-organizing dynamic phenomena*. Oxford University Press, 2016.

[8] ——, "Self-organization and compositionality in cognitive brains: A neurorobotics study," *Proceedings of the IEEE*, vol. 102, no. 4, April 2014.

[9] J. Provost, B. Kuipers, and R. Miikkulainen, "Developing navigation behavior through self-organizing distinctive state abstraction," *Connection Science*, vol. 18, no. 2, 2006.

[10] J. Marshall, D. Blank, and L. Meeden, "An emergent framework for self-motivation in developmental robotics," in *Proceedings of the Third International Conference on Development and Learning*, 2004.

[11] P. Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *Trans. Evol. Comp*, vol. 11, no. 2, pp. 265–286, Apr. 2007. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2006.890271

[12] R. Lee, R. Walker, L. Meeden, and J. Marshall, "Category-based intrinsic motivation," in *Proceedings of the Ninth International Conference on Epigenetic Robotics*, 2009.

[13] D. Blank, "Jyro," https://github.com/Calysto/jyro, 2016.

[14] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[15] "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[16] L. Meeden, "An incremental approach to developing intelligent neural network controllers for robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 3, June 1996.