Paper presented at IASTED NCI Conference, May 19-21,2003, Cancun, Mexico

RGL STUDY IN A HYBRID REAL-TIME SYSTEM

K. Hennacy*, N. Swamy, D. Perlis
Computer Science, University of Maryland
*UMIACS, University of Maryland
College Park, MD 20742, U.S.A.

Abstract

Our work focuses on the study of learning paradigms within a hybrid reasoning system, consisting of a Neural Network and Symbolic Reasoner. There are several aspects to this problem that invite comparisons to work in other fields of study involving for example, knowledge representation, behavioral modeling, and neuromorphic engineering. Central to this paper is a discussion on the coupling between a particular learning paradigm and the behaviors that a real-time system develops in attempting to achieve a specific goal.

Key Words

Emergent Intelligence, Hybrid Reasoning, Real-Time Behavior, Reason Guided Learning

1. Introduction

There has been for some time work in the area of utilizing environmental changes to trigger decision-making in robots [1]. Responses to such triggers have associated with them either explicit or intrinsically represented behavioral models. Behavioral models are not restricted to any particular type of processing. They can be associated with activities such as filtering, prioritizing, monitoring, and reasoning. A particular behavior could involve something as simple as a Boolean test, or as complex as a collection of simultaneous processes that coordinate with each other to achieve a particular result.

Explicit models are identified by those actions of a system (e.g. a robot) that are predetermined by the programming and analysis that is associated with its construction. Whether or not a module exists that is designated as a 'behavior', the 'explicit' classification may still apply. The same goes for whether or not the robot is able to take different courses of action in real-time. If the system takes deliberate steps to process data, and initiate actions, based solely upon instructions that were never learned from experience, its behaviors directly reflect only the intentions, and thinking of its human designer.

We argue that intrinsic models, on the other hand, are ones that are based upon a set of rules for activities that are highly adaptable. Such models give a system the ability to interact intelligently with the outside world without any outside environmental sense initially built into it at all. As the use of the term 'behavior' implies, such a model, as represented by a set of rules and processing activities, is still used for coordinating other activities. However, while the coordination method may be predetermined, the details of the coordination process are not. Instead, they must be learned.

The argument for why learning should be used to distinguish between two broad categories of behavioral models has to do with the recognition that a system's learning prowess is measured by its ability to encode information into a form that isn't specific to a particular scenario—in such a way that what is learned can be broadly and correctly applied. If the encoded information is to originate from experience, then learning is at least implicitly coupled to the behaviors that are responsible for generating the experiences. Thus, we see the emergence of intrinsic behavioral models as being intimately tied to a particular learning paradigm. The deployment of such models will play a key role in developing a system's 'self-awareness', giving it the mechanisms to cope effectively with new, unanticipated situations.

As applied to our study, we are interested in investigating mechanisms that give rise to what we call 'Reasoning Guided Learning' or RGL for short. As the phrase implies, RGL refers to the utilization of goals, (how they are being used, how successful a system is in achieving them, etc.) to determine when and what should be learned. Mechanisms for when learning should take place have so far been investigated within a logic-based framework, with contradictions or time-sensitivity as major triggers for learning [2].

Another important concept associated with RGL is the 'meta-cognitive loop' that must exist to enable the system to increase its competency over time. While reasoning guided learning will improve the system's ability to cope with problematic situations, there is a need to give the system the capability to learn from experiences in order to reason. 'Learning Guided Reasoning', or LGR, thus rounds out the system's ability to gather unstructured information in a context-specific manner, thus providing new information and rules for the system to reason with.

In this paper, we look at the application of LGR to a hybrid system, with the Neural Network initially responsible for making 'decisions'. The inspiration behind the study comes from considering the maturation process of a newborn child. Supposing that the central nervous system is initially in control of the arm or leg motions, over time, as experiences are generated, the neural programming becomes refined and guided at the reasoning, goal-oriented level. Modeling the emergence of such a process, as it applies to the particular case of a robot thus represents part of the research under way to complete the meta-cognitive loop.

The physics-based system under study is a simulated Khepera robot [3], which has two wheels controlling its motion, an array of eight infrared sensors, and a pincher arm that is capable of grasping and lifting objects.

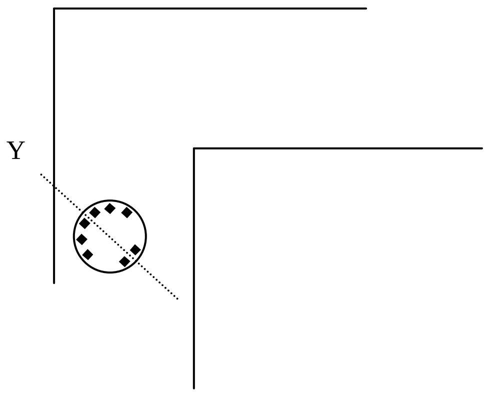

Figure 1.

2. Background

For the purpose of generating cognitive maps, as related to large-scale space, one approach has given robots different levels of instructions and strategies to extract distinct sensor views [4]. Such views are obtained with the help of a hill-climbing technique, for example, to ensure that the robot is able to navigate around corners. Causal graphs are formed to represent the actions that carry the robot from one distinct view to another. In this way, a map of the space is constructed which can be used for navigation.

Such an approach relies upon many 'explicit' behavioral models, chained together with the purpose to perform navigational tasks. While learning is often involved, in determining what may be regarded as 'behaviors', the learning is purposely driven by a planned sequence of actions that build up the capabilities of the robot. While successful, an introduction of intrinsic models into the approach would help the robot to cope with complications associated with varying topologies, moving objects, changes to distinct views, optimization of performance, mistakes due to internal failures or environmental factors, as well as minimize the amount of explicit knowledge that must be programmed to handle these scenarios.

The emergence of intelligence, out of the synergy formed between the cooperation of multiple agents does rely upon intrinsic behaviors to build higher-level behaviors. However, whether manifested within the interfaces of physical robots [5, 6] or simulated components [7], knowledge so far has been shared, or distributed in a way that does not lend itself easily to centralized, high-level reasoning. To facilitate such reasoning, a synergistic (integrated) level goal can be 'negotiated'; however the creation of such goals requires a system capable of learning about its own developing synergies.

There is an extensive amount of investigation being carried out on Brooks' concept-free, situational viewpoint about intelligence, whereby systems can be designed to utilize 'the world as an external memory', requiring little if any symbolic representation for high-level reasoning [8]. Such a viewpoint is bolstered by the use of processing technologies such as neural networks. While logic can be imbedded within neural networks [9], there is an advantage to keeping logical reasoners separate since they are optimized to handle symbolic manipulation. For some time, the integration of neural networks and logic components acting as expert systems has been applied successfully to a variety of tasks such as data mining [10]. However, pure, symbolic manipulation still has advantages for certain technologies, e.g. those involving natural language processing [11].

It is thus worthwhile to investigate the development of abstractions from situational level intelligence as this should represent the grounding of concepts—a necessary criterion for achieving human-level reasoning (represented by activities such as mathematical problem solving, chess, and the creative arts). It is our view that useful insights into this problem can be obtained from a study of an integrated architecture consisting of neural networks and symbolic reasoners designed to create synergistic learning paradigms (within in a real-time environment), since such a topic for research requires a careful consideration of many of the issues raised by the research mentioned in this section.

3. System under Study

For our illustration, will consider the particular case whereby a robot is given the task to solve a navigational problem using RGL. The aspect of this problem we will discuss here involves learning how to avoid collisions with objects.

Consider a fixed neural net topology process, **NN**, which is assigned to control wheel motion:

NN(x, t) = Wheel Velocities

Here, **x** represents sensor data at time ***t***. This function is applied, with periodicity **T**. **T** represents a real-time based, intrinsic processing cycle time for **NN**, irrespective of other processes that may be occurring. An Action Manager is designed to execute a request over a characteristic time τ. After such time, the action is considered completed. We will assume for now that $T > \tau$, to avoid the discussion of what happens when the system processes new information faster than it is able to act upon it.

Hence, the system in this example, goes through a continuous cycle of executing **NN**, sending its result in the form of a request to the Action Manager, and applying the command for a fixed duration of time. The introduction of the Action Manager gives various processing components of the system, such as a logic-based reasoning engine and **NN**, the ability to communicate with each other before a command is sent to the robot's wheels. This is how hybrid reasoning is implemented within this system, with real-time control.

Upon executing the cyclical process, eventually a collision with a wall or object occurs, represented here as a predicate function **C(x)**. There are different ways in which such a system can notice the collision. For this discussion, we will consider the case whereby a peripheral touch sensor is present; however it would be possible to consider the use of meta-reasoning to trigger the awareness, whereby all commands sent to the action manager are monitored for uniqueness (more discussion on this later).

We will associate with the touch sensor, a hard-wired 'knee-jerk' reaction that overrides **NN**, forcing its process to terminate, and causing the robot to back out of the collision according to the following function:

KJ(x, t) = Backward Wheel Velocities

where **KJ** is attenuating over time. When the function no longer has appreciable signal strength, the process associated with it terminates. Note: this is an example of an 'explicit' model that is used to force the robot into a desired state.

After having backed out of the collision, the robot no longer has any processes running to keep it moving. It stops briefly, representing a temporary state of confusion or uncertainty. The Goal Manager recognizes that there are still goals to pursue. One of them represents the persistent desire to move without collisions. For now, we will consider this goal **G** to be hard-wired into the Goal Manager solely for the purpose of illustrating the system's ability to develop strategies for avoiding collisions with other objects. Thus, with the robot stopped, the system immediately recognizes that it failed in achieving its goal of avoiding collisions using the particular **NN** process it had been running.

At this point, a meta-reasoner is invoked to determine what process should be running to achieve the desired goal. In this study, the only process available is **NN**. However, there may be several parameter sets for **NN**, each corresponding to a particular training set that was extracted from the robot's experience over time. Each of these training sets and the **NN** parameters associated with them are referred to here as a particular strategy. Hence, the meta-reasoner determines that process **NN** must be invoked and it must then decide whether to modify an existing strategy, or create a new one, to improve upon the robot's ability to avoid collisions. This is discussed further in the following two sections.

4. Strategy Building

A strategy can be learned in a variety of ways, depending upon the number of permutable elements in the problem characterization, the environmental factors and the experiences received from them at the time of learning; and the activities responsible for processing data and generating an action in response. As such, what strategy is learned will in all likelihood not be optimized. It is, in fact, desirable for this not to be the case. Optimization of a particular strategy should in itself be goal driven, representing a refinement process that may be behavioral driven or deliberately reasoning-guided, based upon the careful observance of experiences.

Hence, within the RGL framework, a strategy **S** is assigned to a particular goal **G** while actually experiencing its successful application to a particular circumstance **C(x)**. This is in contrast to having knowledge beforehand, as expressed in logic, for example, that a desired goal state (in this case, avoiding **C**) will be reached if **S** is applied.

The association, $\mathbf{G[C(x_C)]} \Rightarrow \mathbf{S}$, is formed at the creation of **S**, and is initially represented with a particular sensor state, however the meta-reasoner will attempt to apply **S** again, whenever a collision occurs in a 'similar' fashion to what is characterized by $\mathbf{C(x_C)}$. If a 'similar' situation is not found, then the meta-reasoner decides to create a new strategy, **S'**, based upon its memory of what took place up until the failure.

In our example, a collision strategy is created by training **NN** on a set of (**x** , ***t***) coordinates that it retains up until a collision. The collision point $\mathbf{x}_{C,}$ is replaced by a new point discovered by a trial and error approach. Upon successfully verifying the short term success associated with the new motion, a new training set is created and trained into **NN** using back-propagation. The resulting strategy can be used either as a new approach, for a newly identified circumstance, or it may replace an existing strategy for a group of 'similar' circumstances. A mechanism for deciding between these two options is discussed in the next section.

5. Use of Clusters

The sensor data in our case comes from 8 infrared detectors that are positioned symmetrically about a line (Y) that passes between the pair of wheels (Figure 1). This symmetry can be exploited to identify one particular class of 'similar' circumstances. To illustrate the use of clusters, we will examine the case where the robot is placed into a long, L-shaped hallway.

Upon colliding with a wall the first time, let us say the left wall, a cluster $\mathbf{K_1}$ (formed out of sensor data) is attached to $\mathbf{x}_C$, and is associated with a newly trained strategy $\mathbf{S_1}$. Due to the underlying symmetry associated with the positioning of the infrared sensors, a dual cluster $\mathbf{K_1}$' is formed corresponding to the swap of sensor coordinates across Y. In many circumstances, $\mathbf{S_1}$ will take the robot to the other side of the hall, causing it to collide again.

With the two clusters now already in existence, the meta-reasoner utilizes a distance metric to determine that $\mathbf{x}_C$ lies closest to $\mathbf{K_1}$'. Hence, the meta-reasoner determines that only a permutation of coordinate assignments is required, not a change in strategies. However, to improve upon the strategy, it adopts the method described in the previous section. It also adds a new collision point to the definition of the cluster, triggering the calculation of a new centroid for it.

While this approach will work for the walls of a straight hall, for example, the robot must be able to recognize the need to develop a new strategy when encountering new situations, such as a corner. Ideally, this decision process should also be learned, as the environment may throw in variations that will break not only a strategy, but the recognition of when to use a different strategy.

One approach under consideration involves allowing the robot to remember a sufficient number of collision points to create well-defined clusters. In our illustration, the robot may constantly bounce from the left wall to right wall in the hallway. Upon getting enough points, it will be able to define a sufficient scale for the cluster sizes such that when a corner is eventually encountered, it will be able to determine that a new cluster and strategy needs to be formed.

If a corner is encountered before a sufficient measure of the cluster sizes evolves, an existing strategy may 'incorrectly' be modified. As the robot then encounters new collision points, it will soon find itself failing with circumstances identified within the same cluster, and conclude that the strategy needs further refinement. Upon several updates to its strategies, incorrectly at times, eventually enough of a sample should be obtained for the robot to correctly identify when to update a strategy, and when a new one should be created.

6. Hybrid Learning

A host of other approaches can be taken for the problem we are discussing. In particular, the problem can be solved within the realm of logic. However, to create logic based upon experience, in a way comparable to the method used with the neural network, it is most convenient to have the logic-based reasoner learn from the neural network, and develop new goals and beliefs, before attempting to take over the neural network's functions.

The reason for doing this is not related so much to the desire to have the robot learn how to solve a problem in logic, but rather, to develop a method whereby the knowledge learned within the neural network processor can be raised to a level that involves higher level reasoning—reasoning that is best performed at a symbolic level. Such reasoning could be used in the decision-making for a different class of problems.

Hence, upon allowing the logic based component to learn from **NN**, an ability emerges that allows the logic component to ground the actions of the neural network, associate the actions with its own developed set of commands and predicate functions, and empower it with the knowledge to plan its actions. In our illustration, suppose that a set of successful strategies have been developed by the neural network to avoid collisions. Then, this will mean that as the coordinate space drifts from one well-defined cluster, representing a particular strategy, into another well-defined cluster, the Goal Manager might decide to switch strategies before a collision takes place.

In principle it would be able to do this switching using logic. A meta-reasoner would also be able to notice, periodically, that it is able to refine the robot's existing strategies by taking trajectory points generated by different strategies, and applying them to the same strategy. To perform this operation, however, would require another overriding goal to guide it.

While logic can straightforwardly manage the use of the strategies, in the way they were originally intended to be used, the logical reasoner is capable of learning a more valuable lesson, one that empowers it to utilize the strategies for achieving other goals. Such goals are represented by well-discussed goal states in AI research literature, where for example, the robot desires to move from one end of the hall to another, turn around a corner, or lift an object.

As applied to our example, the logical reasoner will need to have the capability of learning new goal states from experiences, and associating the neural network's collision avoidance strategies towards achieving these goals, if appropriate. To do this will require an exchange between the neural network's knowledge domain, and the logical reasoner's. One way to accomplish this is with the method of coarse-graining. As used here, coarse-graining refers to an activity that is similar to that of the cluster algorithm, only now it is applied to actions, instead of sensor values.

With coarse-graining of **NN** outputs, the logical reasoner is able to create a finite set of commands that characterizes the robot's motion under **NN** control. Upon allowing the logical reasoner to keep track of the use of the neural network strategies, along with coarse-graining of **NN** outputs, the robot will be able to learn command chains, or coarse-grained strategies, that approximately replace the actions of their **NN** counterparts.

The synthesis of command chains involves various levels of precision, determined by the resolution of the coarse-graining algorithms applied to both the actions and the sensor inputs. In our illustration, the resolution of outputs does not have to be high; hence the amount of detail incorporated within a particular coarse-grained strategy will be low. However, to define goal states that the logical reasoner can verify against, coarse-graining will have to be applied to the inputs.

Upon enacting a complete, coarse-grained replacement, the logical reasoner will then be able to establish goal states that are repeatable, either with the use of its coarse-grained actions, or the application of **NN** processing. In this way, a mapping between the **NN** strategies, and a set of chained, coarse-grained logic-based strategies will be achieved. By introducing these goal states, the logical reasoner will be able to utilize memory to form new action-consequence pairs that can be used for logic-based planning.

Chaining, and the goal states associated with them, also introduces new uses for the existing **NN** strategies. With the addition of time measurement, the robot will be able to use its goal states to measure its own performance. Such metrics, along with other goals that encourage 'purpose' into activities (via 'rewards') encourage the refinement of existing **NN** strategies. With this back-and forth interaction, between the logical reasoner, and **NN** processing, the logical reasoner will eventually be able to form new beliefs based upon its observations of the robot's experiences.

For example, in order to reach a desired location as quickly as possible down a straight hall, in order to experience the reward of catching a moving object, the robot should be able to learn, relatively quickly, that the velocities of both of its wheels needs to be set to the same speed.

7. Conclusion

In this paper, we have discussed one approach, within a particular real-time hybrid architecture scheme, of giving a robot the capability of learning new strategies, and modifying existing ones—via a neural network and clustering algorithm. To utilize these strategies as part of higher-level reasoning performed at the symbolic level, it has been argued that it is essential to not only give the logical reasoner the ability to manage the utilization of the strategies; it is essential to give it the capability to examine its experience in applying them, through the use of coarse-graining methods.

References

[1] A. Mali, On the evaluation of agent behaviors, *Artificial Intelligence*, 143, 2003, 1-17.

[2] M. Anderson, Y. Okamoto, D. Josyula, and D. Perlis, The use-mention distinction and its importance to HCI, *Proceedings of the Sixth Workshop on the Semantics and Pragmatics of Dialogue*, Edinburgh, UK, 2002, 21-28.

[3] K-Team, S.A. Switzerland, www.k-team.com.

[4] D. Pierce and B. Kuipers, Learning to Explore and Build Maps, *Proceedings of the Twelth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, Washington, 1994, 1264-1271.

[5] A. H. Cohen, C. Koch, G. Indiveri, R. Douglas, S. Shamma, T. Sejnowski, and T. Horiuchi, Technical Report, *2002 Workshop on Neuromorphic Engineering*, Telluride, CO, 2002.

[6] M. Vona and D. Rus, Self-reconfiguration Planning with Compressible Unit Modules, *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, Detroit, MI, 1999.

[7] H. Bojinov, A. Casal, T. Hogg, Multiagent control of self-reconfigurable robots, *Artificial Intelligence*, 142, 2002, 99-120.

[8] R. A. Brooks, Intelligence without representation, *Artificial Intelligence*, 47, 1991, 139-159.

[9] Boon Toh Low, Reasoning about Beliefs: An Inference Network Approach, *Ph.D. Thesis*, University of Sydney, Australia, 1994.

[10] V. Ciesielski and Gregory Palstra, Using a Hybrid Neural/Expert System for Data Base Mining in Market Survey Data, *Proceedings of The Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Pgs. 36-43, AAAI Press, 1996.

[11] D. Perlis, K. Purang, D. Purushothaman, C. Andersen, and D. Trum, Modeling time and meta-reasoning in dialogue via active logic, *Working notes of AAAI Fall Symposium on Psychological Models of Communication*, 1999.

Acknowledgement: Our appreciation to Alejandro Rodriguez for pointing out the Khepera robot example.