Multirepresentational Architectures for Human-Level Intelligence:
Papers from the AAAI Fall Symposium (FS-09-05)

Conservative and Reward-driven Behavior Selection in a Commonsense Reasoning Framework

Benjamin Johnston and Mary-Anne Williams

University of Technology, Sydney
Broadway, Ultimo, New South Wales 2007
Australia

Abstract

Comirit is a framework for commonsense reasoning that combines simulation, logical deduction and passive machine learning. While a passive, observation-driven approach to learning is safe and highly conservative, it is limited to interaction only with those objects that it has previously observed. In this paper we describe a preliminary exploration of methods for extending Comirit to allow safe action selection in uncertain situations, and to allow reward-maximizing selection of behaviors.

Introduction

Comirit is an open-ended framework that performs multi-representation hybrid reasoning for commonsense reasoning. Comirit combines simulation, logical deduction and machine learning though a unified mechanism that is based upon a generalization of the method of analytic tableaux.

In our previous work, we viewed autonomous learning as principally a 'passive' or 'background' activity. The system continuously observes the environment, attempting to predict future states, and uses prediction error to update its internal model of the world. This mode of learning is integrated as a greedy search algorithm that is implemented by further extending the tableaux reasoning algorithm to prioritize branches of the tableau (*i.e.*, hypotheses about the environment) according to their error.

In this paper, we describe preliminary work that considers how those same mechanisms that enabled the integration of passive machine learning, may also be adapted both to directing action in reward-driven environments and towards a more active mode of learning and behavior. We apply the same greedy algorithm used for passive learning, but generate conservative initial hypotheses from more widely sampling prior knowledge, and adapt the mechanisms for selecting error minimizing hypotheses towards the selection of actions with higher expected rewards.

We begin this paper with a review of the Comirit framework and its motivation. We then consider a variety of approaches to integrating active learning and reward-maximization into the existing framework, and conclude with an overview of future directions.

Background

The original development of Comirit was inspired by the state of modern animations, simulations and computer games. Modern blockbuster computer games offer realistic open-ended 'sandbox' environments for unlimited experimentation and interaction with compelling breadth and depth. While expertise and knowledge is only expressed implicitly in these systems, they appear to lack the brittleness and expense associated with explicit formalization of commonsense in logical languages. We wondered, therefore, whether it would be possible to exploit simulation as a resource of commonsense knowledge, and as a mechanism for commonsense reasoning.

While simulation is a powerful heuristic for deducing possible and likely future states from current conditions, a weakness of simulation is that it must follow the 'arrow-of-time'. That is, it is impossible to simulate a complex situation in *reverse* to deduce likely causes or precursors of a situation: one cannot simulate spilled milk in reverse to discover a likely cause—it is far easier to simulate the outcome from a particular cause such as dropping an open milk carton onto the floor. When greater deductive power is required than that of forward-running simulations, it is necessary to augment simulation with other mechanisms.

Comirit is therefore a proposal to extend simulation-based reasoning into a multi-representational framework. The framework has an open-ended architecture that (so far) combines simulation, formal logical deduction and cumu-

lative learning into a unified mechanism based on the automated theorem proving method of analytic tableaux (Hähnle 2001). This single unifying principle, based on the idea of searching through spaces of possible worlds, enables these disparate mechanisms to be harmoniously combined in a single system.

In the remainder of this section, we provide an overview of simulation, hybrid reasoning and learning. More detailed analysis and explanations may be found in our prior publications (Johnston and Williams 2007; 2008; 2009).

Simulation

In the Comirit framework, simulations are used as the underlying mechanism and representation for large scale commonsense knowledge. Not all knowledge can be represented efficiently in simulations (*e.g.*, 'What is the name of the Queen of England?'), but simulation works extremely well in problems governed by simple laws (such as physics) and so simulation is used in the framework wherever possible.

In the framework, simulations are a sophisticated generalization of an early proposal by Gardin and Meltzer (1989); extended to support 3D environments and non-physical domains. Comirit simulations are constructed from a graph-based representation. The fundamental structure of a problem is first approximated by a (relatively) static graph. The graph is then annotated with frame-like structures, and simulation proceeds by the iterative update of the annotations by functions that perform update.

This representation is intentionally generic. We expect that it can be used to represent simulations from any rule-driven problem domain including physical, social, legal, economic and purely abstract realms. To date, our research has emphasized physical reasoning and naïve physics, so we will illustrate simulation and learning through examples based upon 3D simulations used by physical models.

Consider a simple domestic robot facing a physical reasoning problem: *given a mug filled with coffee, is it 'safe' to perform fast movements to carry the mug?* In the Comirit framework, the robot considers the problem by internal simulations of the scenario; testing whether a simulated mug is damaged by fast movement, or if such motion causes damage to the environment by spilling coffee.

The robot uses a generic graph structure is used to represent the underlying structure of the problem. For example, a mug of coffee can be approximated as a mesh of point masses connected by semi-rigid beams depicted in Figure 1. Examples of particular annotations and values that drive a simulation also appear in Figure 1.

Simulation proceeds by the iterative update of annotation values. Newton's laws of motion are applied to each

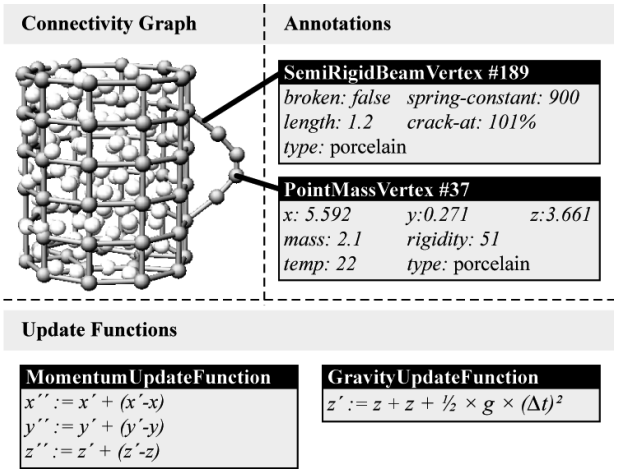

Figure 1: *Examples of the components of a simulation*

of the point masses, and Hooke's law (describing the behavior of a spring) is applied to the connecting beams. Figure 1 illustrates update functions for the laws of momentum and gravity. Note that these functions have only short-term and local effects.

The combined effect of iteratively computing local updates on the annotations is emergent behavior that closely resembles the actual behavior of real world scenarios. Indeed, this method of simulation may be seen as a variation on the Euler method of numerical integration. Analysis of the outcome of a simulation is performed by simple inspection primitives that test the state of simulation to determine if, for example, any semi-rigid bars have broken (in the case of a '**broken**' symbol), or if any liquid is no longer contained by the mug (in the case of a '**mess**' symbol).

Simulation + Reasoning

Recalling that simulation only supports a 'forward chaining' inference mode, we integrate simulation with logical deduction in a hybrid architecture in order to combine the strengths and complement the weaknesses of each mechanism. That is, we use the deductive power of a general-purpose logic to make up for the inflexibility of simulation.

In combining simulation and logic, our experiences are that the conceptual mismatch between simulation and logic prevents the application of traditional integration techniques such as blackboard architectures. Our attempts invariably resulted in systems that were unworkably complex and difficult to maintain. Instead, we sought a clear and unifying abstraction to harmonize the semantics of the reasoning mechanisms; by interpreting both simulation and logical deduction as operations that manipulate spaces of possible worlds.

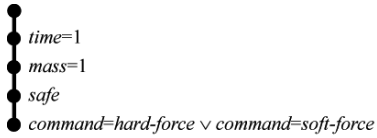

Figure 2a: *First, each conjunct in the original query is expanded into separate nodes.*

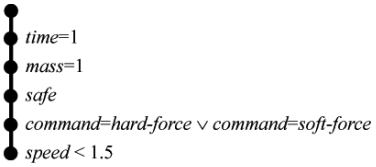

Figure 2b: *The term 'safe' is expanded per its definition.*

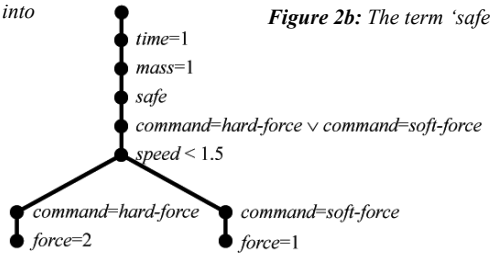

Figure 2c: *The tableau is forked into two branches: one branch for each disjunct in the fifth node. 'Hard-force' and 'soft-force' are then expanded per their definitions. Logical deduction has now stalled: no more logical rules apply.*

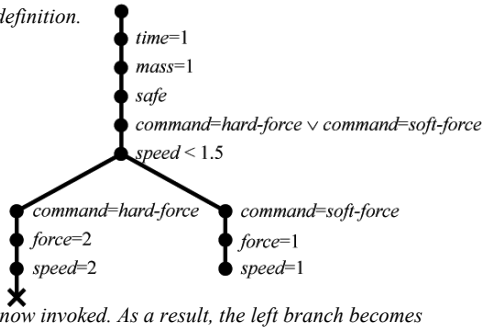

Figure 2d: *Simulation is now invoked. As a result, the left branch becomes inconsistent (speed = 2 and speed < 1.5). The branch remains open and therefore describes a scenario satisfying the original query (i.e., the robot can safely use soft-force).*

The method of analytic tableaux (Hähnle 2001) is an efficient method of mechanizing logical theorem proving. Analytic tableaux have been successfully applied to large problems on the semantic web, and there is a vast body of literature on their efficient implementation (*ibid.*). The method constructs trees (tableaux) through the syntactic decomposition of logical expressions, and then eliminates branches of the tree that contain contradictions among the decomposed atomic formulae. Each branch of the resultant tableau may be seen as a partial, disjunction-free description of a model for the input formulae. The crucial insight is that if a tableau algorithm is given knowledge of the world and a query as logical input, then the conjunction of the atomic formulae along a branch of the resultant tree represents a *space of worlds that satisfy the query*.

The tableau method *and* simulation can thereby be unified through this common abstraction. The tableau algorithm generates spaces of worlds, and simulation expands upon knowledge of spaces of worlds (*i.e.*, forward chains to future states based on current states).

In our framework, we perform commonsense reasoning by generalizing the tableau so that it can contain non-logical terms such as simulations, functions and data-structures in addition to the standard logical terms of traditional tableaux. Integration is achieved by mapping the mechanisms of diverse reasoning modes onto the tableau operators for expansion, branching and closing of branches. Traditional logical tableau rules map without change, and simulation is treated as an expansion operator.

To illustrate this mechanism, consider the following scenario:

> *A household robot needs to move an object across a table. Its actuators can perform a soft or a hard movement. It is unsafe to move any object 'quickly'. What commands may be sent to the actuators?*

For the convenience of our example calculations, let us assume that the mass of the object is 1kg, the soft force is 1N, the hard force is 2N, the object is simply pushed for 1s and unsafe speeds are $1.5ms^{-1}$ or higher. Furthermore, we assume the following highly simplified and abstracted simulation (while simplified, this simulation has the characteristic properties of being numerical, fully specified and only operating in a 'forward direction'):

```
function simulate(Mass, Force, Time):
  set Speed := Time * Force / Mass
  return {speed = Speed}
```

We can then convert the scenario to a logical form:

time=1 ∧ *mass*=1 ∧ *safe* ∧
(*command=hard-force* ∨ *command=soft-force*)

With this logical form, we may then apply the method of analytic tableau and simulation to find models that satisfy the formula. The algorithm proceeds in the steps illustrated in Figures 2a–2d, terminating with only one open branch. Reading atomic formulae along that remaining branch, we see that it describes a world in which the action, *command=soft-force* is applied and the object moves safely at $1ms^{-1}$.

Thus, we have used both simulation and logical deduction in a single mechanism to solve a (simplified) commonsense reasoning problem. In our prototypes, we incorporate a range of optimizations and special techniques: rule selection heuristics, rule for meta-reasoning, indexes, and unbound Prolog-style output variables.

Simulation + Reasoning + Learning

Of course, an intelligent system is of limited use if it has neither knowledge nor experience. While our expectation is that the acquisition of knowledge will be partially supported by manual engineering, our plan is that the system will autonomously acquire knowledge through observation of (and interaction with) the world.

A key advantage of simulations is that they are easily constructed and configured from observations. For exam-

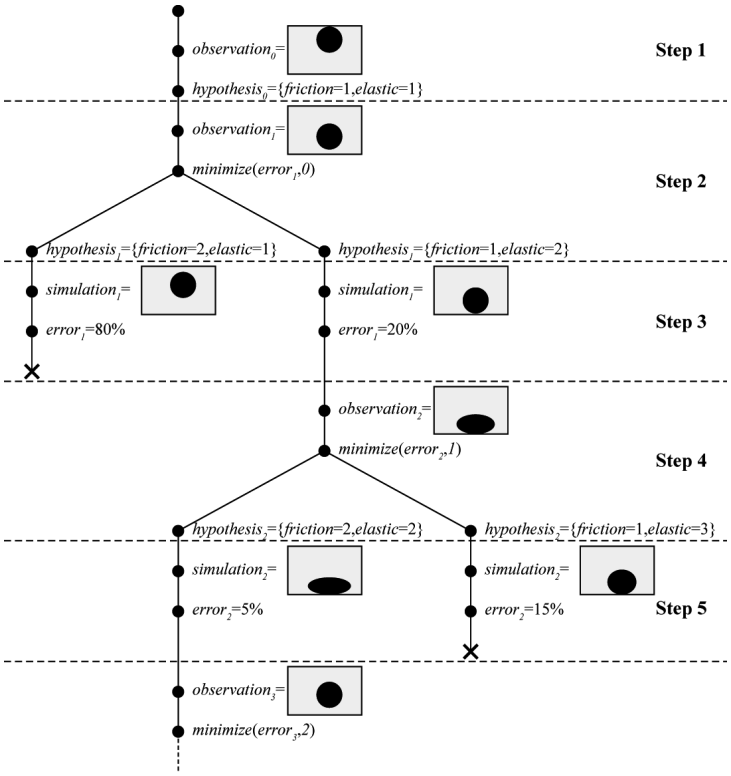

Figure 3: *Examples of the components of a simulation*

ple, given the ability to sense an object's 3D shape (such as by stereographic cameras, laser scanners or time-of-flight cameras), an accurate simulation can be built from just two or three observations (Johnston and Williams 2009). The 3D structure is scanned directly to create a graph approximation of the object, and visual observations of behavior are used to assign values to the annotations on the graph. Contrary to intuitions, a single pair of visual observations is sufficient to achieve significant accuracy: each pixel in an image is a separate data-point, so that two images may provides many thousands of training samples for learning the precise parameters of a simulation.

In Comirit, machine learning is implemented by mapping hypotheses generation and selection onto the tableaux reasoning rules. Generating hypotheses is akin to the disjunctive branching of the tableau, and the evaluation of hypotheses and subsequent discard of poor hypotheses is akin to closing branches in a tableau due to inconsistency.

This is achieved by introducing an (incomplete) ordering over branches, and then modifying the tableau algorithm so that it searches for minimal models. A branch in a tableau is no longer considered 'open' simply if it is consistent per the traditional tableaux algorithm: it is open if it is either *consistent and unordered*, or else *it has an ordering and is minimal among all other consistent and ordered branches*. That is, the algorithm considers all unordered branches, and only one ordered branch.

We define the ordering over branches using symbols that are stored *within* the tableau. The set of propositions **`minimize`**(*VariableName*, *Priority*) are assumed to be tautologically true in any logical context, but are used for evaluating the *order* of the branches. The *Priority* is a value from a totally-ordered set (such as the integers) and indicates the order in which the values of variables are sorted: branches are first sorted by the highest priority variable, then equal values are sorted using the next highest priority variable, and so on.

Consider our household robot example again. If the robot encounters a novel object (say, a ball) it can then use a stochastic hill-climbing strategy on its observations of the object in order to build an accurate simulation of the object, as depicted in Figure 3:

Step 1: The tableau initially contains the first observation of a ball and the initial hypothesis generated (many other control objects, meshes, functions and other data will be in the tableau, but these are not shown for simplicity).

Step 2: The system observes movement in the ball. It generates new hypotheses by random perturbation, seeking to find a hypothesis with minimal error.

Step 3: The system simulates from $hypothesis_0$. The result of the simulation is compared with $observation_1$ to determine the error in the hypothesis. The right branch has smaller error so the left branch is no longer open.

Step 4: As with Step 2, the system observes more movement and generates new randomly perturbed hypotheses, further refining the current hypothesis.

Step 5: The system then continues as with Step 3, but this time the left branch is minimal. In the following steps, the algorithm continues yet again with more new observations and further hypothesizing…

Note also that because our tableaux can contain logic, simulations and functions, the system may use logical constraints or ad-hoc 'helper functions' even when searching for values in a simulation (*e.g.*, a constraint such as $mass > 0$, or a custom hypothesis generator that samples the problem space in order to produce better hypotheses).

Directed Learning

In our work to-date (as illustrated above), learning is performed as a passive activity of observation and belief update. The system, in a background process, constantly anticipates future outcomes and compares expectations against observations to update internal simulations. However, passive observation is clearly not sufficient for a general purpose system: an object at a stable equilibrium does not yield useful data for learning. In order to build accurate simulations, the system must observe the dynamic properties of objects. That is, a system must actively engage with the environment in order to construct accurate models of the environment.

Consider a domestic robot facing the typical home in which virtually every object lies at rest, in static equili-

brium. It is not reasonable to expect that every new object must be taught to the system (not only is there the vast range of products that may be bought by an occupant, but also the possibility of entirely home-made objects). Instead, the robot needs to independently learn how to safely interact with *any* object.

A number of factors complicate learning in such environments: the *most informative* action is likely to be destructive (*e.g.*, one way to determine whether a glass is fragile would be throw it at the wall), and yet there does not appear to be a single set of informative actions that can be performed on all objects (consider the different forms of safe interaction for a 'house of cards', a glass of liquid and a large cardboard box).

Hypothesis Generation

In Figure 3, we used a simple two-element hypothesis to illustrate learning the properties of a ball. In practice, the hypothesis is a large vector of numerical and non-numerical parameters that corresponds to the full set of annotations of an object's simulation. Several iterations of the greedy stochastic search are applied for each observation, allowing fast learning and convergence on these large vectors.

In fact, we previously proposed that the hypothesis space of the learning algorithm should include a large Self-Organizing Map (SOM) (Kohonen 1998) of background knowledge of all objects it has encountered. The SOM helps the system generalize its knowledge, because similar values cluster around each other in the map: resulting in faster learning, and better initial hypotheses about new objects.

Put simply, a self-organizing map is an auto-associative neural network formed from a 2D grid (or map) of vectors. A training vector is 'learnt' by the network by searching for the most similar vector presently in the network (the 'winner'), and then adjusting the values of that vector (and the other nearby vectors on the map) to be closer to those of the training vector. The rate of learning is a parameter of time: initially, each training sample initially results in a large adjustment of the 'winner' and its neighboring vectors, but as the model converges diminishing updates are applied over a diminishing neighborhood.

When a new object (in stasis) is first encountered, the system will be able to directly observe superficial properties: its size, its color and texture, its visual similarity to other familiar objects, and whether the object appears to be intricately constructed or appears simply as solid block of matter. The system should use this external knowledge as cues to guessing safe behaviors. For example, a shiny 'metallic' cup is likely to be far more robust than a fine porcelain tea-cup embossed with intricate patterns. The self-organizing map can make use of the visually observed properties by including visual attributes on the training and model vectors. When a novel object is first encountered, the map is searched for a vector whose immediately observable values most closely match the new observation: that 'partial winner' can then be used to provide a good initial hypothesis for the unobservable parameters that must be learnt. Given this initial hypothesis, simulations can then be run to evaluate the range of possible actions that may be performed; and select an action that will lead to more learning or an external reward.

More formally, the 'partial winner' of an input vector v is chosen as the vector x from the set of model vectors M in the SOM as given by:

$$\underset{x \in M}{\operatorname{argmin}} \|(v - x)F\|$$

where F is the diagonal matrix diag(c_1, c_2, …, c_n) such that c_i is equal to 1 if the ith element of the vector is directly observable and 0 otherwise.

Note that the computation of error in selecting a 'partial winner' may also be measured using non-Euclidian and weighted metrics as appropriate for a given problem domain.

Conservative Reasoning

A risk of selecting the single hypothesis that is the 'partial winner' in a self-organizing map is that appearances can be deceptive. For example, it is difficult to distinguish between: a hot skillet versus a cold skillet; a mould of unset Jell-O versus set Jell-O; and a cardboard box of fragile objects versus solid objects. We prefer a robot to act conservatively until it is certain that it can act safely.

So, while the direct selection of a 'partial winner' provides for excellent initial hypotheses (Johnston and Williams 2009), these hypotheses represent the 'best guess' rather than a 'most conservative guess' of the object's behavior. Instead, we propose that the robot encountering an unknown object in stasis should sample plausible hypotheses, and defensively presume the worst outcome.

Thus we propose extending the initial hypothesis, so that it is not a single 'partial winner', but a set of the top-N 'partial winners'.

That is, the initial hypothesis for an observation v, is given by a set of N vectors X of such that, $|X| = N$ and is subject to:

$$\max_{x \in X} \|(v - x)F\| \leq \min_{x \in M - X} \|(v - x)F\|$$

Each vector in this set of 'partial winners' is used to instantiate separate simulations that together sample the full spectrum of likely effects of action on the object. A conservatively safe action is one that has the *best possible worst-case behavior*. If the robot's hypothesis of a cardboard box contains N vectors, where simulation with some vectors suggests that the cardboard box is solid and robust, while other vectors that suggest that the object may be fragile; then the robot will select the gentle action that has the best worst-case behavior.

Learning-oriented Activity

While we have promoted the selection of activities based on optimal worst-case performance, this is not the only option available to a system. For example, if an agent's goal is simply information gain; then actions may be chosen based upon their ability to maximally discriminate between the vectors in an initial hypothesis.

We would suggest, however, that while this approach may maximize the rate of learning and may be ideal for an 'infant robot' in a controlled environment, it is not likely to be suitable in real-world contexts. Smashing an object against a wall can be highly informative, even though it is unproductive. Indeed, such action is not even necessary, since good simulations can be learnt with very few observations.

We would suggest that an agent with a well-trained self-organizing map does not need to perform significant active learning upon encountering a new object. The greatest uncertainty lies only with objects at rest in static equilibrium. Sampling the SOM to select a conservative action is sufficient for initial interactions, and the result of this action will generate new observations for rapid learning and model refinement.

Thus, we view active learning as principally a process of selecting a safe initial interaction. A top-*N* set of 'partial winners' should be used to select a best worst-case action on an unknown object, and then the standard 'winner'-takes all strategy of a SOM is used for subsequent observations, behavior-generation and learning for that object. More sophisticated learning in highly uncertain, dynamic or dangerous environments can proceed by sampling (and updating) a top-*N* set of 'partial winners' with subsequent decay in the size, *N*, of that set.

Goal-Directed Activity

Integration of passive learning into the Comirit framework uses **`minimize`** terms. While these terms were intended to allow the selection of cases (or models) that contained the best hypotheses, the same mechanism finds application in reward-directed activity selection. Where learning is a process of postulating models of the world and discarding those models with poor match to the observed world; goal directed behavior can be similarly modeled in Comirit by considering each action in a separate case (or tableau branch) and discarding those cases with sub-optimal expected return.

That is, goal-directed activity is modeled in Comirit by: instantiating the set of possible actions as separate cases; instantiating **`minimize`**(*VariableName*, *Priority*) terms; setting corresponding variables with the *negation* of the expected reward. The tableaux algorithm will automatically close minimum valued (*i.e.*, maximal reward) branches. When only one branch remains and that branch contains an action, the robot (or software agent) then executes the action.

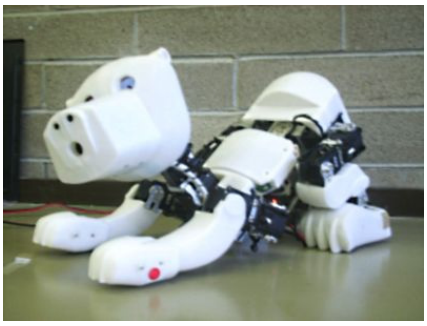

Figure 4: *The Hykim robot bear.*

Conclusion and Future Work

In this paper we have described how mechanisms for passive learning are readily adopted to: performing conservative actions in an uncertain environment; and goal-directed activity. These ideas are preliminary—they describe work in progress which has not been fully tested—and so we view this work as an 'extended position statement'.

We are presently developing the framework on the Hykim robot bear (see Figure 4). The robot runs the Debian operating on an Intel-compatible PC, it has 21 degrees of freedom, and a range of sensors including a high-resolution web-camera (mounted in the snout) and a WiFi network adapter. Our ambition is to create a robot with innate curiosity and a sense of play: it will explore its own environment to discover and learn about everyday objects, and apply the knowledge that it learns towards solving problems. It will use the multi-representational framework described in this paper to learn arbitrary physical and non-physical systems, and then use its understanding of those objects to act towards its goals. Our long term view is for robot systems that perform large scale autonomous acquisition of commonsense knowledge, that can perform in *any* environment, and that serve as a platform for exploring deep questions of embodiment and symbol grounding.

References

Gardin, G. and Meltzer, B. 1989. 'Analogical representations of naïve physics', Artificial Intelligence, vol. 38, no. 2, pp. 139–159.

Hähnle, R. 2001. 'Tableaux and Related Methods', Handbook of Automated Reasoning, vol. I, pp. 100–178, Elsevier Science.

Johnston, B. and Williams, M-A. 2007. 'A generic framework for approximate simulation in commonsense reasoning systems', Proceedings of COMMONSENSE 2007, pp. 71–76.

Johnston, B. and Williams, M-A. .2008. 'Comirit: Commonsense reasoning by integrating simulation and logic', Proceedings of AGI-2008, pp. 200–211.

Johnston, B. and Williams, M-A. 2009. 'Autonomous Learning of Commonsense Simulations', Proceedings of COMMONSENSE 2009, pp. 73–78.