A Computational Theory of the Communication of Problem-Solving Knowledge between Parents and Children

Ph.D. Proposal

Bo Morgan
MIT Media Lab
20 Ames Street
Cambridge, MA 02139 USA
bo@mit.edu

Bo Morgan
Student of Media Arts and Sciences, MIT

Joseph A. Paradiso
Professor of Media Arts and Sciences, MIT

Marvin Minsky
Professor of Media Arts and Sciences, MIT
Professor of Electrical Engineering and Computer Science, MIT

Gerald J. Sussman
Professor of Electrical Engineering and Computer Science, MIT

Michael T. Cox
Program Manager, DARPA

A Computational Theory of the Communication of Problem-Solving Knowledge between Parents and Children

Ph.D. Proposal

Bo Morgan
MIT Media Lab
20 Ames Street
Cambridge, MA 02139 USA
bo@mit.edu

***Abstract*—We propose to develop a theory for how a social culture of how problems are solved is learned and communicated between generations in a species. Our presentation and validation of our theory includes a description, a model implementation, and a quantitative as well as qualitative evaluation. Our theory of how children learn from parents is based on a problem-solving algorithm spanning a society with a combination of experienced and inexperienced agents. Cultural knowledge is a powerful form of transmitted knowledge that sets the state of the inexperienced individual to an approximation of the optimal state for helping the individual solve problems. It is important for computational theories of learning to provide explanations for (1) learning by trial and error, (2) learning by mimicry, as well as (3) learning by being told. This last form of learning by being told is important because it allows for much of the power of human cultural knowledge to be used by machines. Our theory is aimed to show how an understanding of learning by trial and error can be generalized to multiple layers in a reflective critic-selector learning algorithm in order to provide one cohesive theory for all three of these forms of learning. We present a relational social blocks world simulation with children and parents that will be used to evaluate an implementation of our theory by a collection of human subjects who provide their commonsense evaluations of the emotions and behaviors expressed by our model in the context of different example narratives.**

***Keywords*-common sense reasoning; cognitive architecture; critic-selector architecture; metareasoning; credit assignment; programming language; reflective causal tracing; self-models; imprimer models**

I. Introduction

We have focused our thesis on the parent-child relationship because this is the first form of social relationship that an individual experiences. Parents and children share special and powerful forms of communication, based fundamentally upon scold-guilt and praise-pride command languages [1]. We model social life as a system of evolutionary survivability in a world of limited resources in which a set of individuals compete for these resources in order to reproduce. Because our mathematics incorporates simple genetics and sexual reproduction, our primary social relationship is the parent-parent-child sexual reproduction relationship. Other secondary forms of relationship in our model include the parent-stranger and child-stranger relationships. In terms of genetic competition and individual survivability, we compare the properties of resource distribution with these most basic relationship structures. The individuals in our model are rewarded for solving problems that are distributed in a simple spatial world.

In order to make these problems similar to the types of problems that humans solve in common everyday situations, we focus on a problem-solving domain within a simple 3D commonsense computer game environment. Problem-solving in an individual problem solver has been well studied, but how a society of problem-solvers develop has not.

II. Hypothesis

In terms of well established problem-solving metrics in AI, we may have time to piecewise test aspects of our architecture against various standards from the more narrowly focused fields of computer science and machine learning.

In our model, parents and teachers sometimes provide positive or negative feedback to our model of a child. So, if time permits, we will circumscribe this small aspect of our model for comparison against a modern relational reinforcement learning problem [2] (e.g. reinforcement learning in a relational blocks world domain).

Also in our model, we can consider part of our reflective layer as implementing what is referred to as "chunking" in the explanation-based fields of machine learning [3] and cognitive science. So, if time permits, we can compare the efficiency of reflective "chunking" in our system with our system without this part of our reflective layer.

We hypothesize that each of the reflective layers in our cognitive model enable our algorithm to handle incrementally more complex problem-solving domains. We will use a social relational problem domain to quantitatively show how our (1) reflective, (2) self-reflective, and (3) self-conscious layers of our algorithm affect performance. We expect the complexity of the specific relational problem

domain to affect performance, so we will evaluate the goal-state attainment performance of our algorithm in a number of commonsense scenarios of varying complexity.

In addition to quantitative problem solving metrics, it is important to measure the correlation of activations of technical pieces of our algorithm with commonsense folk psychological states of mind [4]. We expect that some parts of our algorithm will correspond with commonsense psychology terms, such as "surprise", "disappointment", "guilt", "pride", etc. For example, Figure 5 shows questions that address the commonsense nature of the six layers of our model in the context of solving a problem in an example scenario.

III. Background

1) Common Sense Requires Many Ways to Think: An architecture that has only one way of thinking will get stuck on some types of problems, so we are focusing on building an architecture that can switch between multiple ways of thinking. These ways of thinking are connections and activations of constellations of reflective, deliberative, and reactive sets of critics and selectors. The original theory for how to build such an architecture was published in [5] as a model involving layers of self-reflective, self-conscious, and self-ideal critics for reflective control of an already reflective algorithm. Also, the reader is referred to this work for a more detailed description of the critic-selector algorithm. We need an architecture that allows many different ways to think because while a single reasoning method may work for a specific problem, no any one reasoning method works for solving all different types of problems [6]. Perhaps one day we will discover a simple key or mathematical formula for deriving all other intelligence, but let's first focus on building a machine that demonstrates any kind of robust commonsense intelligence before making any simplifying optimizations. What we see as necessary for developing a model of commonsense intelligence is the ability to quickly switch between different representations of problems that allow different reasoning methods to continue where any single method would have gotten stuck. See [7] for an overview of the Panalogy architecture for a more detailed explanation of why multiple representations that invoke different procedures of thought are necessary for building a model of a robust problem solving mind.

A. A Review of The Emotion Machine v1.0

One system that implements commonsense reasoning, based on Minsky's Emotion Machine theory of mind [8], is a metareasoning system for correcting faulty plans, called EM1 (Emotion Machine, v1) [9]. EM1 is written in Lisp, using a Prolog extension as the logical resolution tool. EM1 is a layered architecture consisting of reactive, deliberative, and reflective layers. Mental critics are represented as commonsense narratives that result in queries to a collection of different Prolog knowledge bases. The commonsense narratives are given to the system in a Lisp format that is compiled into the knowledge bases as collections of horn clauses. These knowledge bases consist of collections of domain-specific horn clauses that are divided into physical, social, and mental domains of reasoning. On top of this Prolog logical substrate, the Lisp program is organized into layers as a critic-selector model of mind [10]. The narrative plans that are generated by the deliberative layer are executed by a lower-layer, called the reactive layer. Part of the reactive layer of the algorithm is written in C and runs PID control loops in a simulated social two-wheeled inverted pendulum type robot. EM1 demonstrates how a system can use commonsense narratives in order to reason by analogy in order to generate plans. Also, EM1 demonstrates a learning process that is driven by reflective critical recognition of failure. Because of the complexity of the rigid-body physics in the world, sometimes even the most carefully constructed plans fail. EM1 has a layer of reflective critics that debug deliberative narratives as they are being interpreted by using a collection of commonsense narrative debugging critics. Using narratives about social situations, EM1 infers the goals of the other agents in the world given partial knowledge of their visible physical actions. When mistakes are made in this inference process, the failure is recognized reflectively, after the fact. Specific types of debugging responses are implemented for different forms of critical failures. EM1 is a step toward a large and complex commonsense reasoning agent with multiple layers of metareasoning that inspect, control, and debug mental representations.

B. Closed-loop Control and Learning as a Cognitive Model

There are many AI algorithms that provide explanations for how to accomplish goals or gather rewards in a domain. A basic AI system consists of three processes: (1) perceptual data are generalized and categorized to learn induced abstract models, (2) abstract models are used to infer expected hypothetical states, i.e. states of future, past, or otherwise "hidden" variables, (3) actions are chosen based on considerations of different action-dependent inferences.

While there are many types of machine learning algorithms that focus on this abstract 3-step closed-loop process of learning to control, the field of meta-cognition [11] focuses on making at least two layers of closed-loop systems. The first closed-loop learning algorithm learns how to deal with the external world, while the second closed-loop learning algorithm perceives the state of the algorithm below. We see meta-cognition as a layering of learning algorithms, such that the second layer algorithm learns from perceiving the activity of the first layer and controls or modifies this first layer. While it may be clear how to trace changes in the perceptual inputs of layer one of the algorithm, it is less than clear how the second layer learner should monitor the changes in the state of the first layer learner.

	Phy	Rea	LR	Del	Ref	SRef	SC
Roboverse [12]	X	X					
LifeNet [13]			X	X			
Act-R [14]			X	X			
CogAff [15]			X	X	X		
EM-1 [9]			X	X	X		
Funk2 [16]			❁	❁	❁		
Commonsense-S	❁	❁					
Moral-C			❁	❁	❁	❁	❁

X — The model implements this reflective layer.
❁ — The model will implement this reflective layer.

Figure 1. **Modelling projects for AI and what reflective layers they have included:** physical layer (Phy), reactive layer (Rea), learned reactive layer (LR), Deliberative Layer (Del), Reflective Layer (Ref), Self-Reflective Layer (SRef), Self-Conscious Layer (SC). Commonsense Simulator is the environment for which our Moral Compass AI model exists as a controller.

C. The Theoretical Space of Reflective Cognitive Models

Figure 1 shows how different cognitive models have focused on implementing specific ranges of the reflective layer stack. Roboverse is a rigid-body physics simulation that implements physical laws and also has basic goals for a robot, such as moving to a location, or picking up an object. Roboverse is a good example of a model that is not an AI model. Roboverse is a physics model. Every AI model is as opposed to a physics model. Control theory would call the physics model, the model of the plant, while it would call the AI model, the model of the controller. Our AI model consists of five layers of control algorithms that handle different types of goals: (1) Learned Reactive, (2) Deliberative, (3) Reflective, (4) Self-Reflective, and (5) Self-Conscious. These are roughly the same layers as Minsky's Model-6 theory. CogAff is one of only a few reflective cognitive architectures being researched in the world today, and they have focused on modelling three of six of the Model-6 layers of reflection. EM-1 is the first example of a critic-selector architecture with three layers of reflection modelled after the bottom three layers of the Model-6 theory. Funk2 is our reflective programming language project that will allow us to easily build an arbitrary structure of layers of reflective control. Moral Compass is a cognitive architecture that includes Reflective, Self-Reflective, and Self-Conscious critic-selector layers that allow an agent to act realistically in a physical social simulation. Moral Compass uses the Commonsense Simulator as the physical model that it is controlling and thinking about.

D. The Layers in our Model

Physical actions are special types of agent actions because they are actions that, unless visually obstructed, are observable by other agents. Mental states of other agents can then be inferred by analogical comparison to an agent's own memories of the mental states that caused similar physical streams of actions. A few examples of physical goals that an agent can perform in the commonsense simulation are as follows:

1) move to
2) move in direction
3) set posture
4) move near object
5) pick up object
6) drop object
7) use object with object

Because our model is primarily meant to model the internal workings of a person's mind, it contains additional layers of mental goals that influence the pursuit of basic physical goals. The deliberative layer of goals makes plans for how to serialize and parallelize the basic physical goals. The deliberative layer implements a form of planning based on the critic-selector model. Examples of deliberative goals in our model include:

1) infer successful action meant to cause state
2) infer failure action meant to cause state
3) recall similar state
4) recall similar successful action
5) infer plan to goal state
6) recall similar plan causing state

Deliberative goals sometimes run into problems themselves, so an additional reflective layer of goals exists in order to coordinate when and how deliberative thought should progress. Examples of a few reflective mental goals include:

1) infer plans conflict
2) infer plans share serial resource
3) combine two plans
4) check plan action effect clobbers action precondition bug
5) ignore deliberative goal
6) focus on deliberative goal
7) recall similar successful plan
8) recall similar failure plan
9) induce debugging transframe[1] between plans
10) apply debugging transframe to plan

Even a deliberative layer with reflective control of its processes can run into problems. In order to keep track of what types of problems that a system is good or bad at solving, it uses "self-models" that keep track of its strengths and weaknesses. Goals that use these self-models in order to provide further insight into how the deliberative and reflective layers should be trained, modified, or otherwise controlled, we refer to as self-reflective goals. Examples of a few self-reflective mental goals include:

1) activate reflective personality
2) suppress reflective personality

[1]**transframe:** a frame object representing a number of changes to be made to a frame's slots that results in another frame. Transframes are used here to represent changes in relational (i.e. frame-based) representations.

3) check personality focus/ignorance conflict
4) recall similar successful personality
5) recall similar failure personality
6) induce debugging transframe between personalities
7) apply debugging transframe to personality
8) recall similar successful personality accomplishing focus/ignorance goal

Examples of a few self-conscious mental goals include:

1) infer cause of agent imprimer guilt
2) infer cause of agent imprimer praise

E. A Programming Language Specifically Designed for the Next Emotion Machine

Funk2 [16] is a programming language that we have written with the following programming language and cognitive architectural goals in mind:

1) causal tracing of arbitrary lisp-like processes,
2) massively concurrent multi-threaded simulations, including many thousands of lightweight parallel processes,
3) layers of reflective critic-selector problem solving,
4) commonsense reasoning cognitive architecture primitives, such as person, event, goal, belief, narrative, etc.

We have completed goals one and two in our current described implementation. We are currently working on goals three and four.

Just after Push Singh completed his PhD work of building EM1 in 2006, he sadly passed away, leaving a large project filled with great potential for future research, but since 2006, no one to our knowledge has worked on any other critic-selector cognitive architectures related to this work. Because of this, there seems to us to be a hole in this field of reflective critic-selector cognitive architecture research. Two years ago this fact was very clear to us, and we at that point volunteered for the job of rewriting the original architecture in order to pursue some of the original goals. We have programmed the beginning of a new Emotion Machine critic-selector architecture. As there are many future directions for the EM1 architecture, there were also many problems with the architecture as it was implemented then. Some of the primary concerns that we have with this original implementation are as follows:

- Reasoning is very slow with only 21 narratives in the memory of the architecture.
- Critics and selectors are specified in a declarative logical form, which does not allow for reasoning over noisy or imperfect data.
- The declarative form of the Critic-L language allows for inserted procedural Lisp code, but any procedural code inserted in this way is not reflectively traced.
- Critics and selectors cannot run in parallel, which eliminates the potential for using the architecture for solving any form of parallel control problem.

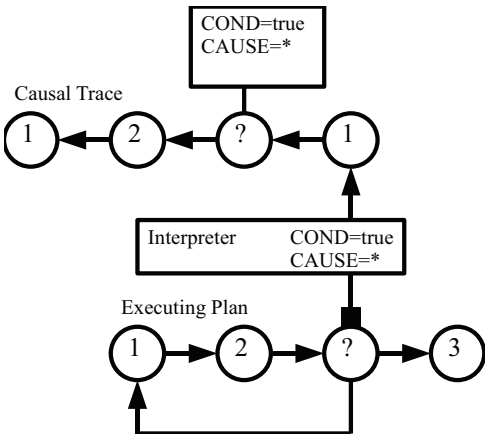

Figure 2. An example of an executing plan containing a loop being interpreted and leaving a trace of plan events with conditional events tagged with contextual information, including the condition evaluated (COND) as well as the causal context (CAUSE). The symbol "*" refers to a list of goals, activated critics and selectors, the function call stack, and other user-specified contextual hints. The "?" represents a conditional expression in a "while" loop.

- The architecture has great potential for parallelization but cannot take advantage of multi-core CPUs or distributed processors.
- Architecture is based on the expensive proprietary commercial languages of Allegro Lisp and Allegro Prolog, which cannot be used freely by researchers.
- Self-reflective, self-conscious, and self-ideal critics and selectors are future work to still be implemented.
- Critics and selectors are not learned from experience.

1) Tracing Procedural Branches in Causal Context: Figure 2 shows a schematic example of how a plan interpreter scans along a plan and, whenever it encounters a conditional branch, leaves a causal trace of reasons for each decision made by the process interpreting the plan.

Causality refers to an agent that causes a change in the world. In our case, the agent is vague and consists of the current context of the computation; for example, what goals are currently actively being pursued. This type of memory is important to keep track of in many different representations itself, so that when a change is recognized, perhaps long after the fact, as a bug, then this information can be used for considering multiple different debugging strategies, each associated with a different representation of the cause. For example, all decisions in any computer program can be thought of as a machine code instruction that checks whether specific register is equal to zero in order to decide whether or not to jump to another part of the program. All decisions in a computer are executed at this level of detail, but this is not a good representation for high-level thinking about causality. For most of our causal representations we would prefer a more abstract description of the situation or event in which the decision took place, what agent or agency

within the mind was responsible for the code that made the decision. In any case, whether we are debugging a low-level or high-level idea, we prefer to err on the side of being able to record more information than less. Causal tracing can always be set to only trace specific layers of abstraction in Funk2, so machine code tracing is only relevant if the system wanted to reflectively learn something about it's own low-level implementation. Typically, we imagine users of Funk2 mainly tracing and building critics to recognize patterns in their own user-defined procedural events. There are many other parts of a causal relationship that we would like to be able to refer to in a reflective process that is debugging a bug in a plan:

- Event or situation.
- Agents or agencies involved.
- Knowledge used for compiling those agents.
- Critics and selectors that were active during the compiling of this decision point.
- Reflective processes that were monitoring the planning process at the time.
- Self-reflective critics and selectors that were being used to control the reflective focus.
- Configurations of self-models activated or suppressed during plan creation.
- Goals that were active.
- Self-conscious critics and imprimers[2] that were active, if any.

This a short list of the types of causal knowledge representations that must be handled by different types of causal tracing critic agents.

F. Funk2 Represents each Critic and Selector as a Separate Thread

Funk2 is an abstract simulation of a distributed operating system. In addition to the normal primitive data-types of most popular languages, Funk2 also allows access to the following primitive data-types as first class citizens:

- cause
- funktion
- object-type
- fiber (virtual thread)
- scheduler
- processor

Also, like many modern languages, Funk2 represents all primitive and abstract data within the language as frames with named and typed slots. Built-in slot types for frames include *get*, *set*, and *execute*. The object system is relatively primitive thus far, without incorporating much of the meta-object protocol [17], but we expect our unique *cause* object to take care of most of the previously complicated meta-object protocol by putting the protocol into the evaluator. The Funk2 core is written entirely in C, just like all popular operating systems these days, so any operating system or external package specific extensions to the language should be easy additions.

1) The Cause Object: Cause objects are created and linked to parent cause objects with the creation of every new execution event, such as the spawning of a new thread. Typically, tracing of multi-threaded programs gets complicated because of the complex inheritance structure of causes for events. Because we have designated a special register in each thread for a cause object, which monitors and controls memory access, we expect to handle this problem more efficiently by always having this abstract control at the locus of every memory access. Without causal tracing of memory access invoked, our interpreter runs at full speed, while when a piece of memory that requires causal tracing encounters a traced cause, this will create events that are appended to a cause-specific trace. We expect cause objects to be one simplifying key to many complicating problems of credit assignment.

Cause objects are frames with typed slots, just like all data in Funk2, but because cause objects follow the causal execution paths of processes, these objects can be used for storing different types of traces of process executions. Funk2 is a reflective frame-based programming language, in which case, "an object would not only represent information about the thing in the domain it represents, but also about (the implementation and interpretation of) the object itself: when is it created? by whom is it created? what constraints does it have to fulfill? etc." [18].

G. Implementation of Mental Resources as Funk2 Fibers

We refer to critics and selectors as types of "mental resources" in our cognitive architecture. As previously discussed, in the EM1 Critic-L implementation of the Emotion Machine architecture, these resources are specified in a lisp declarative form. Because we wish our architecture to remain representationally agnostic and thus more generally applicable for a variety of programmers working together, our mental resources are implemented using any form of the lisp-like Funk2 code. For example, for every mental resource, we allocate a virtual process that we call a fiber; a fiber is to a Funk2 program as a thread is to a C program. Fibers are scheduled by the Funk2 core and are very lightweight bytecode interpreters that execute in parallel. All processes in Funk2 are executed within fibers. Funk2 allows intricate control and inspection of fibers that allow pausing, resuming, and rescheduling fibers. Each mental resource, such as a critic or a selector, is allocated a separate fiber that is initially paused. The architecture supports thousands of concurrent mental resources that can be activated and suppressed by one another. We imagine that most mental resources in a given critic-selector architecture are not active at any given moment. Each mental resource can execute arbitrary lisp-like Funk2 code, given that it is in an activated state. Further, all

[2]**imprimer:** a mental simulation of a role-model.

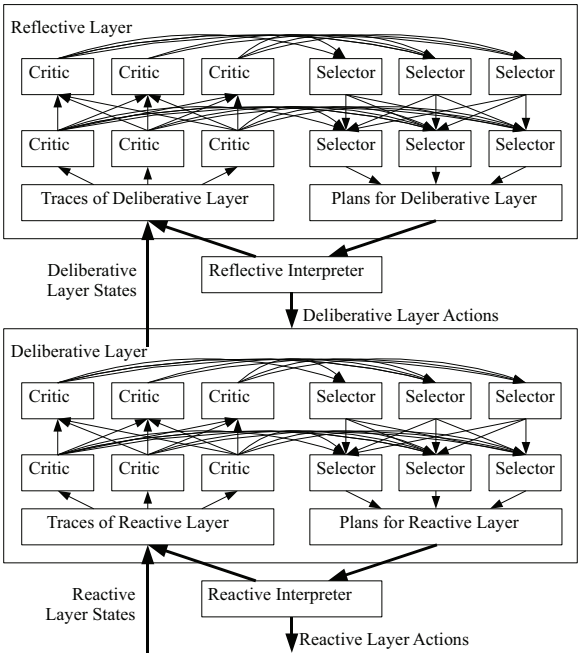

Figure 3. This diagram shows roughly how information flows within and between two of the six layers discussed in the Emotion Machine critic-selector architecture.

data that is manipulated by a fiber is manipulated within the context of the current local cause that has activated the mental resource.

H. How Funk2 Causal Tracing Helps Learning with Critics and Selectors

Figure 3 shows roughly how information flows within and between two of the six layers of the Emotion Machine critic-selector architecture. Unlike in EM1, we expect the next version of this architecture to allow arbitrary procedural descriptions of critic and selector agents, rather than strict declarative logical forms. The added causal memory tracing features of the Funk2 programming language allow credit assignment algorithms to still trace causal agency between critics and selectors. Funk2 keeps track of many types of primitive memory events for all objects in the language, which allows programmers the freedom to express themselves in a way that is natural for someone familiar with languages such as Python, Java, C++, or Lisp. Funk2 inherits the object-oriented ideas from Java, C++, and Python, such as dynamic object frames and object types, while also incorporating the interactive compiler and "data as code" minimalist syntax ideas from Lisp. While we expect to incorporate a few logical programming ideas from Prolog, we will focus our approach to problem of deduction, induction, and inference from more of a commonsense natural language perspective [19].

IV. Theoretical Approach

A. Our Model

The primary feature of our model of life is that the process of life is a social endeavor, so we introduce the set of social individuals, $\mathbf{I}$.

$$I^* = |\mathbf{I}| \tag{1}$$
$$\mathbf{I} = \{I^0, I^1, I^2, ..., I^{I^*-3}, I^{I^*-2}, I^{I^*-1}\} \tag{2}$$
$$I \in \mathbf{I} \tag{3}$$

Also, since life is a physical process, we will now define a model of the physical space, in which our individuals exist. We will use a two-dimensional grid that will be the part of our objective model that will be analogous to the real surface of the Earth. We define each dimension of our two-dimensional spatial world as a variable, $\mathbf{X}$, from the natural numbers including zero and bounded by the supremum, X^*:

$$X^* = |\mathbf{X}|, \tag{4}$$
$$\mathbf{X} = \{0, 1, 2, ..., X^* - 3, X^* - 2, X^* - 1\}, \tag{5}$$
$$X \in \mathbf{X}. \tag{6}$$

The positions, $\vec{\mathbf{X}}$, in our two-dimensional model are defined as the product of two of these dimensions,

$$\vec{\mathbf{X}} = \mathbf{X} \times \mathbf{X}, \tag{7}$$
$$\vec{X} \in \vec{\mathbf{X}}, \tag{8}$$
$$\vec{X} = \{X_x, X_y\}. \tag{9}$$

Each individual, I, has a physical position, $\vec{I_X}$, such that

$$\vec{I_X} \in \vec{\mathbf{X}}. \tag{10}$$

B. The Social Process of Genetic Evolution

Each individual in our model has a genetic binary string, I_G, that determines that individual's phenotypical behavior. We save our discussion of how the genes of an individual determine the phenotypical parts of the individual, let us just say now that these genes determine the initial state, $I_M(t = 0)$, of the individual's "mind" process.

The primary social relationship in our model is one of sexual reproduction, requiring two parent individuals for every child individual. The crossover combination of two gene lines during sexual reproduction allows combining successful mutations from multiple gene lines, an important adaptive speed advantage for sexual over asexual reproducers. The sexual reproduction event has two main parts, a set

of two parents, $\{I_{P_0}, I_{P_1}\}$, and a child, I_C:

$$I_{P_0} \in \mathbf{I}, \tag{11}$$
$$I_{P_1} \in \mathbf{I}, \tag{12}$$
$$I_C \in \mathbf{I}, \tag{13}$$
$$I_{P_0} \neq I_{P_1}, \tag{14}$$
$$I_C \neq I_{P_0}, \tag{15}$$
$$I_C \neq I_{P_1}, \tag{16}$$
$$\mathbf{I_P} = \{I_{P_0}, I_{P_1}\}, \tag{17}$$
$$I_P \in \mathbf{I_P}. \tag{18}$$

Also, in order to clear space for a new individual to exist, we must have individuals die, so that they free positions in the physical world for a new variety of genetic individuals to exist. Every individual has a birth time, I_{t_b}, a life length, I_{t_l}, and a death time, I_{t_d}:

$$I_{t_l} \geq 0, \tag{19}$$
$$I_{t_d} = I_{t_b} + I_{t_l}, \tag{20}$$
$$t \geq I_{t_b} \text{ and } t < I_{t_d} \Leftrightarrow [\text{alive } I]. \tag{21}$$

C. Environmental Energy Sources

In our model, we create every individual with a personal amount of energy storage, like a battery. The ability of a individual to survive until death from old age becomes a question of the ratio of energy used over energy gathered from these puzzles.

We introduce a limited life-giving resource, without which an individual is dead. We call this life-giving resource "energy," and we represent energy as a variable from the natural numbers including zero.

$$\mathbf{I_E} = \{0, 1, 2, ...\} \tag{22}$$
$$I_E \in \mathbf{I_E} \tag{23}$$

We define being dead for an individual as that individual having no energy, and we define being alive as not being dead.

$$I_E = 0 \quad \Leftrightarrow [\text{dead } I], \tag{24}$$
$$I_E \neq 0 \quad \Leftrightarrow [\text{alive } I]. \tag{25}$$
$$\tag{26}$$

D. Energy Providing Puzzles

These energy providing puzzles can be thought of as games that one can choose to play, which results in either winning or losing. For example, one type of puzzle would be playing a game of chess against an artificial opponent. Another type of puzzle would be a block stacking problem in the relational "blocks world" domain. Yet another such type of puzzle may take place in the Commonsense Simulation, which we will describe shortly.

An energy providing puzzle has a set of possible states, $\mathbf{P_S}$, where each such state, P_S, is a labelled directed graph. We choose a labelled directed graph as our state space because this mathematical object can represent any data structure in a frame-based computer program, so our games include any game that can be represented as a computer program. We now define a "natural" Markovian temporal function, $P_{\mathcal{N}}$, for the puzzle state. Each puzzle also has a set of action functions, $\mathbf{P_A}$, which an individual, I, can cause to occur. Each of these action functions has a number, P_{A_n}, of arguments. Some of these actions may be "perceptual", which like functional processes, have a "return" result but otherwise do not affect the system. Some of these actions may be "motor," which has a physical effect on the puzzle object, possibly resulting in a change in the agent's energy, I_E, and possibly also returning a value. The overall transition function, $P_{\mathcal{T}}$, for the puzzle is not Markovian and depends upon an individual's mind process to choose a puzzle action, $I_{P_a}(t)$, and a list of arguments, $\mathbf{I_{P_r}}(t)$, which together fully specify the individual action puzzle transition function, $I_{P_{\mathcal{A}}}(t)$.

$$I_{P_a}(t) \text{ and } \mathbf{I_{P_r}}(t) \Leftrightarrow I_{P_{\mathcal{A}}}(t), \tag{27}$$
$$P_{\mathcal{T}}(t) = P_{\mathcal{N}} \circ I_{P_{\mathcal{A}}}(t). \tag{28}$$

This type of puzzle process is similar to the popular problem in AI of learning to control a Markov Decision Process (MDP) for which there are very few solutions for the large state and action space of arbitrary computer programs that we have defined here. The observation, $I_{P_o}(t)$, of the individual is a subset of the state, $P_S(t)$, of the puzzle.

$$P_{S_0} \in \mathbf{P_S}, \tag{29}$$
$$P_S(0) = P_{S_0}, \tag{30}$$
$$P_{\mathcal{T}}(P_S(t)) \in \mathbf{P_S}, \tag{31}$$
$$P_S(t+1) = P_{\mathcal{T}}(P_S(t)), \tag{32}$$
$$I_{P_o}(t) \subseteq P_S(t). \tag{33}$$

Any type of problem, whose domain can be represented as a labelled directed graph, can be used as a puzzle in this sense. These types of problems, we call a "puzzle" and are often in very large spaces, $|\mathbf{P_S}| \gg 0$, with transition functions, requiring $|\mathbf{P_S}|^2$ bits in general to specify. These calculations assume that we are only working with graphs with a finite number of nodes and edges.

Puzzles exist at a physical location, $\vec{P_X}$, such that

$$\vec{P_X} \in \vec{\mathbf{X}}. \tag{34}$$

If an individual exists at the same position as a puzzle, we call that individual, the "solver", P_I, of the puzzle:

$$P_I = I \Leftrightarrow \vec{P_X} = \vec{I_X}. \tag{35}$$

E. The Commonsense Puzzle Simulation

Figure 4 shows a simple puzzle simulation that is based on a three-dimensional world drawn with simple two-dimensional images. Our world is interesting not because

Figure 4. **Commonsense Simulator:** A simple commonsense simulation, based on a bounded continuous three-dimensional world with predefined objects and actions. This world is similar to the classic "blocks world" simulation, except that it is meant to model human learning in complex social narratives that include children, parents, and strangers. Commonsense Simulator exists as one of the puzzles that an individual, I, is meant to reason about and control in order to gather energy.

of the large number of states that it physically simulates but instead because of its ability to simulate complex social narratives with which our agent interacts. The puzzle world contains multiple individuals, some children, some parents, and some strangers that can communicate with one another. However, only the puzzle solver, P_I, can perform actions in order to interact with the puzzle, resulting in a change in energy, P_{I_E}, for the puzzle solver.

V. Evaluation Method

As with any theoretical modelling project, a question of evaluation comes down to the question: how well does the model correlate with reality? Because our theory has many variables in the reflective layers that control a social human being during learning, there are many ways that we can think of potentially evaluating the realism of our model in a study. For example, as our agent is simulated in our simple physical commonsense simulation, we can ask human subjects whether or not the performance of the AI agent is realistic. For each layer of the algorithm, we can ask survey questions, beginning with the lowest, physical, layer. For example, Figure 5 shows questions that address the realism of the six layers of our model, the physical simulation as well as five reflective layers in our AI algorithm, namely the following layers: (0) physical simulation, (1) physically reactive, (2) deliberative, (3) reflective, (4) self-reflective, (5) self-conscious.

VI. Expected Contributions

Our contributions will include the following:

1) the Commonsense Simulator for creating simple physical and social narratives,
2) the Funk2 open-source reflective programming language,
3) the Moral Compass model-6 critic-selector cognitive architecture,
4) a catalog of implemented reflective, self-reflective, and self-conscious critics,
5) a human experimental study of the commonsense realism of the different processes in our implementation of this social theory of learning.

VII. Time Line

1) 2010 January: Demonstrate Funk2 deliberative layer knowledge and planner.
2) 2010 February: Demonstrate reflective category learning from plan failure.
3) 2010 March: Demonstrate learning of self-reflective models by clustering reflective behaviors.
4) 2010 April: Demonstrate learning of values of self-reflective models by parental praise and scolding.
5) 2010 May: Evaluate realism of model with user study.
6) 2010 June: Finish writing dissertation.
7) 2010 August: Defend dissertation.

References

[1] W. Meyer, "Therapy of the Conscience: Technical Recommendations for Working on the Harsh Superego of the Patient," *Clinical Social Work Journal*, vol. 26, no. 4, pp. 353–368, 1998.

[2] S. Džeroski, L. De Raedt, and K. Driessens, "Relational reinforcement learning," *Machine Learning*, vol. 43, no. 1, pp. 7–52, 2001.

[3] G. DeJong, "Explanation-based learning," in *Computer Science Handbook*, 2nd ed., A. Tucker, Ed. Chapman & Hall/CRC and ACM, 2004, pp. 68.1 – 68.18.

[4] A. Gordon and J. Hobbs, "Formalizations of commonsense psychology," *AI Magazine*, vol. 25, no. 4, pp. 49–62, 2004.

[5] P. Singh and M. Minsky, "An Architecture for Combining Ways to Think," 2003.

[6] J. McCarthy, M. Minsky, A. Sloman, L. Gong, T. A. Lau, L. Morgenstern, E. T. Mueller, D. Riecken, M. Singh, and P. Singh, "An architecture of diversity for commonsense reasoning," *IBM Systems Journal*, vol. 41, no. 3, 2002.

[7] P. Singh and M. Minsky, "An architecture for cognitive diversity," *Visions of mind: architectures for cognition and affect*, p. 312, 2005.

[8] M. Minsky, *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. New York, New York: Simon & Schuster, 2006.

[9] P. Singh, "EM-ONE: an architecture for reflective commonsense thinking," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

The following questions refer to Carol, the little girl in **Example Narrative 1**, as shown in the problem-solving simulation on your computer screen. Please answer the following questions in terms of how realistic it is for the mental or physical event to occur at this point in the example narrative.

Physical Simulation Realism

1.	The mud object slips through the tines of the fork.	least realistic	0...1...2...3...4...5...6...7...8...9...10	most realistic

AI Reactive Layer Realism

1.	Carol is playing by herself in the mud.	least	0...1...2...3...4...5...6...7...8...9...10	most
2.	Carol picks up the fork object.	realistic	0...1...2...3...4...5...6...7...8...9...10	realistic
3.	Carol walks to the mud object.		0...1...2...3...4...5...6...7...8...9...10	
4.	Carol uses the fork object to lift the mud object.		0...1...2...3...4...5...6...7...8...9...10	
5.	Carol puts down the fork object.		0...1...2...3...4...5...6...7...8...9...10	
6.	Carol picks up the shovel object.		0...1...2...3...4...5...6...7...8...9...10	

AI Deliberative Layer Realism

1.	Carol wants to put the mud in the bucket.	least	0...1...2...3...4...5...6...7...8...9...10	most
2.	Carol remembers using a shovel to pick up mud in the past.	realistic	0...1...2...3...4...5...6...7...8...9...10	realistic
3.	Carol finds a fork and a shovel to be similar because they both have square scoopers.		0...1...2...3...4...5...6...7...8...9...10	
4.	Carol decides to try to use the fork to pick up the mud.		0...1...2...3...4...5...6...7...8...9...10	

AI Reflective Layer Realism

1.	Carol stops focusing on trying to put the mud into the bucket when the stranger scolds.	least	0...1...2...3...4...5...6...7...8...9...10	most
2.	Carol is surprised by the stranger scolding her.	realistic	0...1...2...3...4...5...6...7...8...9...10	realistic
3.	Carol chooses to focus on the goal of finding safety.		0...1...2...3...4...5...6...7...8...9...10	

AI Self-Reflective Layer Realism

1.	Carol is scared for her physical safety by the presence of the stranger.	least	0...1...2...3...4...5...6...7...8...9...10	most
2.	Carol thinks that she is physically small and weak compared to the stranger.	realistic	0...1...2...3...4...5...6...7...8...9...10	realistic

AI Self-Conscious Layer Realism

1.	Carol starts crying because she is ashamed of getting mud on her skirt.	least realistic	0...1...2...3...4...5...6...7...8...9...10	most realistic

Figure 5. A questionaire that evaluates the realism of individual states and actions in both the physical simulation and the AI model.

[10] M. Minsky, *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind—Thinking, The Critic Selector Model of Mind*. New York, New York: Simon & Schuster, 2006, chapter 7.2, pp. 220–224.

[11] M. Cox and A. Raja, "Metareasoning: A manifesto," *BBN Technical*, 2007.

[12] B. Morgan, "Roboverse: Physical Robot Simulation," 2003.

[13] B. Morgan and P. Singh, "Elaborating sensor data using temporal and spatial commonsense reasoning," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, 2006, p. 4.

[14] J. Anderson, D. Bothell, M. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," *Psychological Review*, vol. 111, no. 4, pp. 1036–1060, 2004.

[15] A. Sloman, "Varieties of affect and the cogaff architecture schema," in *Proceedings Symposium on Emotion, Cognition, and Affective Computing AISB*, vol. 1. Citeseer, 2001, pp. 39–48.

[16] B. Morgan, "Funk2: A distributed processing language for reflective tracing of a large critic-selector cognitive architecture," in *Proceedings of the Metacognition Workshop at the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO-09)*, 2009.

[17] G. Kiczales, D. Bobrow, and J. des Rivieres, *The art of the metaobject protocol*. MIT press, 1999.

[18] P. Maes, "Issues in computational reflection," in *Meta-level architectures and reflection*. North-Holland, 1988, pp. 21–35.

[19] H. Liu and P. Singh, "Commonsense reasoning in and over natural language," *Lecture Notes in Computer Science*, pp. 293–306, 2004.