Critic Networks for Commonsense Problem Solving

Heikki Ruuska
Minsky Institute for Artificial Intelligence
Heikki.Ruuska@sci.fi

Abstract

In his book *The Emotion Machine*, Marvin Minsky discusses a layered Critic-Selector model and uses it as a starting point for developing a metatheory of problem solving. One of his starting assumptions is that human problem solving abilities and resourcefulness don't stem from a small number of neat methods for processing uniformly structured data, but from diverse control systems able to coordinate among multiple different processing paradigms and their associated knowledge.

Experiments for implementing such control systems were begun by Push Singh in EM-ONE [Singh, 2005], and continued by Bo Morgan in his SALS AI system [Morgan, 2013]. In our current work, we have done further experiments based on Push Singh's system. Our system is a case-based reasoner which uses commonsense narratives, short stories about the world and mental events, and manages mental Critics, which are abstractions of procedures that recognize types of problems in the world and inside the system and suggest actions to resolve them. The Critics form networks, which contain different potentially useful ways of processing detected problems, and instructions on how to evaluate the results.

The actual processes implemented by these Critic networks can be very diverse, such as problem reformulation, ambiguity detection, planning, communication, and resource allocation. Our early results show promise of the approach towards solving construction problems, and in construction and assessment of commonsense narratives that could be useful for anything that benefits from having knowledge accessible as scripts. Furthermore, the abstract reasoning behind Critics encourages and enables human-readable experiments on cognitive metamachinery.

1 Introduction

Push Singh's *EM-ONE* is a cognitive architecture for reflective problem solving for problems involving physical, social and mental realms [Singh, 2005]. He developed his program to implement parts of Marvin Minsky's Emotion Machine architecture [Minsky, 2006][Minsky *et al.*, 2004].

One of Minsky's starting assumptions is that our problem solving abilities and resourcefulness don't stem from a small number of neat methods for processing uniformly structured data, but from diverse control systems able to coordinate among multiple different processing paradigms and their associated knowledge. Singh's EM-ONE and its derivative described here are attempts to build such control systems.

In Singh's example scenario, two simulated robots collaborate to build a table, using a model of a mind based on mental *Critics*. The networks formed of Critics enable the robots to guess at each other's goals, generate plans for physical problem solving, and reflect on reasons behind failed interpretations.

Singh's work has since been improved on by Bo Morgan. In his PhD thesis [Morgan, 2013], he describes SALS AI, which is a new system resembling EM-ONE that greatly improves the planning and plan-based tracking and reflection aspects.

Singh passed away in 2006; the source code he left behind wasn't in a functional state. We re-implemented his program, using parts of the original code, in two stages: first, a version that faithfully recreates the functionality described in his thesis, and second, a version with new Critics, new knowledge, and new features in the internal languages, as described in Section 3.

The program described here is based on Singh's original Lisp syntax, the core languages of which, *Narrative-L* and *Critic-L*, are highly modular and human-readable. Whereas the languages are strictly symbolic, being context-defined, phenomenom-based descriptions, they allow for ambiguities that can be resolved by heuristics or arbitrary data. The modular nature of Critics and loose interpretation of symbols also allows them to be used for decision processes in conjunction with non-symbolic representations, such as neural nets, provided sufficient translation protocols are constructed.

Mental Critics written in Critic-L can modify each other based on their experiences. This enables them to recognize new types of dangers and opportunities, and alternative ways for planning to achieve the higher-order goals of the program they are a part of.

Since many of the Critics are higher order Critics which operate on other Critics, it follows by design that the program can not only describe its reasoning symbolically, but also on different levels of abstraction and at different points of a causal chain. This makes it possible to both write fixes for special cases, and introduce entirely new methods of reasoning, without having to retrain the whole system or risk breaking existing functionality.

The next section overviews the original core components of EM-ONE, including the database system, the *Narrative-L* and *Critic-L*, and discusses their implementation in the program.

Section 3 discusses current and in-progress improvements to EM-ONE, and technical features they depend on. These features are new and don't exist in Singh's thesis.

2 Original Components

2.1 Terminology

In Minsky's Critic-Selector model [Minsky, 2006], Critics are problem patterns, and Selectors potential solutions to dealing with the problems by activating selected mental resources. Singh's use of the term Critic is a special case of a Critic with an optional Selector, referring both to abstracted problem patterns and also to productions containing consequent actions, which make use of the instantiated antecedent-side patterns. In this paper, we keep with Singh's usage. An example Critic is *difference-between-conditions-and-desires=¿propose-action*, which detects an unfulfilled goal and proposes a default action.

The main components of our program are a knowledge representation language, Narrative-L, and a process representation language, Critic-L. By design, knowledge expressed in these languages can be either manually authored, or constructed from other sources.

Implementation-wise, both Narrative-L and Critic-L are compiled into Prolog-like collections of facts and rules. The facts are stored in a database. The Critic clauses typically match against items in the database and, based on the results, activate other Critics, assert new clauses into the database, or modify existing Critic networks.

2.2 The Fact Database

A major form of pattern matching between Critics is carried out by a purpose-built Prolog-like subsystem built on top of Common Lisp. The syntax of the Prolog system resembles Norvig's [Norvig, 1991]. The Prolog system relies on a fact database, where knowledge is stored in the following format:

```
(item [database] [item-name]
  ([predicate] [value1] [value2]) [factid])
```

database the database name, e.g. narratives, conditions, hypotheses
item-name identifier, such as current-conditions
predicate a predefined predicate, such as ACTOR
value1 a slot value, usually a situation id, such as SIT_1234
value2 a slot value, such as GREEN

The **(predicate value1 value2)** constructs define semantic subgraphs, which are indexed by the **database** and **item-name** identifiers. The subgraphs are treelike structures where hierarchies are defined by special **subsit** predicates, which take situation id's in both of their slots.

The names of all binary predicates, and their possible slots, are defined on the system side of the program. However, the actual semantics of the basic predicates are dependent on userspace data, which consists of Narrative and Critic definitions, a description of the world state, and generated hypotheses.

2.3 Narrative-L

Narratives are a primary knowledge representation in the system. They are storylike descriptions of physical, social and mental events and relations. They consist of predicates and various slots. Whereas the syntax of each predicate is strictly defined by its pre-assigned slots, their semantics and pragmatics are left to be defined by the contexts they are used in.

The current world state is stored in the database of binary predicates described in the previous section. Narrative-L is a language parsed into that representation. Narrative-L narratives are lists of clauses consisting of predicates and frame slots. An example narrative:

```
(defnarrative attaching-stick
 (sequential
   (observes pink (not (is-attached stick board)))
   (does pink (attaches pink stick board) [1])
   (observes pink (is-attached stick board) [2]))
 (causes [1] [2]))
```

We have divided narrative predicates into distinct types by how they compile into the binary representation: basic, group and supplementary. Basic predicates, such as *observes* and *is-attached*, generate binary predicates based on their frame slots, such as *actor, prop* and *subject*. Group predicates, such as *sequential*, group basic predicate situation-ids together by generating binary predicates to join basic predicate situation id's under a higher-level situation-id with *subsit*, and can attach temporal relations between basic predicate situations with the *follows* binary predicate. Supplementary predicates, such as *causes* and *jointly*, add additional semantics that Critics can make use of, and don't generate additional situation id's of their own.

The narrative parser automatically assigns situation id's to represent each basic and group predicate, and in case of nested predicates, expresses the relationship with the *subsit* predicate. It generates appropriate frame slot predicates based on predefined templates. The result of parsing is a collection of Prolog facts. An example of the generated semantic subnet is depicted in **Figure 1**.

2.4 Critic-L

Critics are case-based pattern-matching procedure rules. The Critics in EM-ONE can have an antecedent and consequent part, where the antecedent part is typically a collection of patterns to be matched to, and the consequent part a sequence of actions to take if all the patterns match. It's also possible to have a Critic with only one of these parts.

Critics can produce hypotheses, elaborate on them, and give orders for actions to be executed in the outside world. They typically act in networks; they are distinct processes which are specialized to detecting or dealing with a particular type of a problem. The defining feature of EM-ONE is the existence of Critics which manage and debug the behavior of other problem-solving Critics. These Reflective Critics

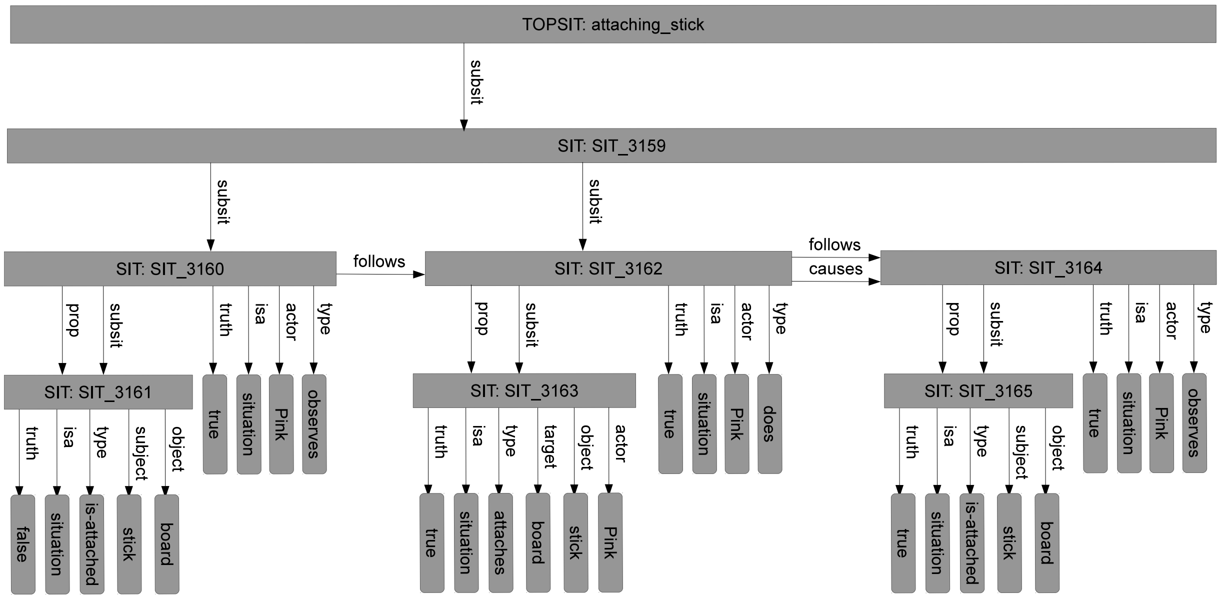

Figure 1: Expansion of a short commonsense narrative.

are invoked when, for example, a plan didn't work out or produced undesired or unexpected side-effects.

Critics are authored in Critic-L, which is a language similar to Narrative-L. Patterns in the Facts database can be accessed through an **in** construct, as in:

```
(in narratives ?N
 (desires :actor ?ACTOR
          :prop (is-attached :subject ?S
                             :object ?O)))
```

This generates a matching network (**figure 2**) where the values of matching variables are all candidates for later use through backtracking. Unlike in Narrative-L, the desired frame slots need to be explicitly labeled: besides implementational reasons, this is to allow partial matches to clauses without the need to write in filler variables. Another way to access knowledge embedded in narratives is through Extractors, which are commonly occurring patterns and can be used to declutter Critic definitions.

To be able to infer action consequences and make plans, Critics need ways to represent possible future situations. This is done through the *hypotheses* database, which is a subset of the Fact database and shares the structure, with the exception that hypothesis identifiers of the form H_1234 and N-SIT_1234 are used alongside the SIT_1234 form. As a consequence of the similar form, the hypotheses can also be accessed using the same **in** construct.

Critics can generate hypotheses by copying over relevant parts of the world state, the narrative knowledge base, or other hypotheses. They can then edit the hypothesis, typically based on what they predict might happen, or might have happened, according to the patterns in the Critic body. Finally, appropriate Critics analyze hypotheses, and special ranking and actuator Critics choose appropriate actions or record beliefs.

Another feature of Critic-L is the initiation and control of processes to assess hypotheses; currently this is done by running the antecedent parts of Critics included in an assessment network and constructing a summary of matching patterns. Highest level (executive) Critics generally accept the least-criticized hypotheses as desirable plans of actions, recording the intents and extracting the first actions to be carried out.

Since Critics consist of lists of clauses of predicates, they can be easily modified on the fly. In particular, if a Critic notices that a wrong conclusion has been made due to an incomplete assessment network, it can amend the Critic controlling that network. Or, if a Critic notices that another one is too generic, matching to so many situations that its criticisms are essentially worthless, it can disable the ailing Critic; furthermore, experimental Critics can try removing or adding clauses in other Critics to see how that affects their predictions.

Examples of Critics can be seen in sections 3.1, 3.2 and 3.6.

3 Improvements and Additions

In this section, we discuss some improvements we have made to Singh's original EM-ONE and how these help it function in additional domains.

3.1 Application Languages

EM-ONE Critic networks depend on having the knowledge represented in the binary predicate format. Usually, conversion is managed by the Narrative-L parser, but it is possible to make interfaces for mining snippets of knowledge from existing commonsense knowledge databases, such as ConceptNet [Havasi *et al.*, 2007] or CyC [Lenat, 1995].

For example, we can generate mini-Narratives out of ConceptNet relations such as **(MadeOf stick wood)** by a pattern:

```
(narrative-constructor implies-ingredient
```

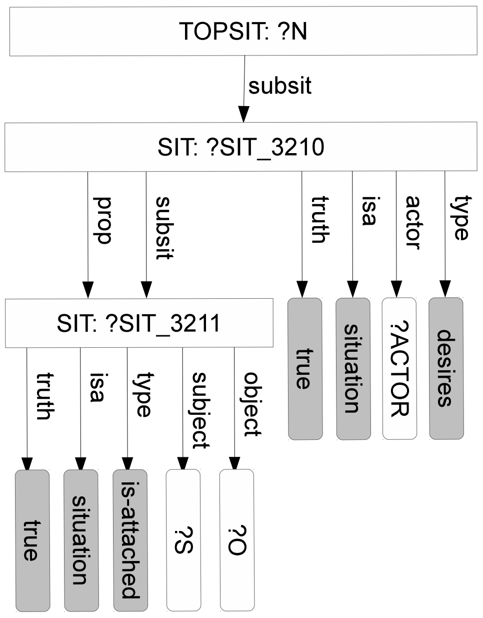

Figure 2: A network generated by a Critic-L *in* operator.

```
 (conceptnet-clause (MadeOf ?OBJ ?MAT))
 (or (conceptnet-clause (IsA ?MAT material))
     (conceptnet-clause (RelatedTo ?MAT material)))
(=>)
(make-narrative
  (make-name ingredient-requirement- ?OBJ ?MAT)
 (observes actor ?MAT [1])
 (does actor (convert-object actor ?MAT ?OBJ) [2])
 (requirement [1] [2]))
```

Similarly, we can mine natural language texts for more Narratives, by using such commonly occurring patterns that don't usually generate very ambiguous results. Parsing narratives from natural language uses specialized Critics which generate parsing hypotheses. The hypotheses are ranked and evaluated, with high-scored hypotheses being accepted. The following disambiguates a personal pronoun. Note that this Critic doesn't check for gender: potential nullifying Criticism is administered by a separate Critic, which searches for gender mismatches.

```
(defnlpcritic (same-sentence-actor-pronoun=>disambiguate)
  (in parsing-hypothesis ?P1
   (in-sentence ?S
    (occurs-only-once :slot actor ?ACTOR [1])
    (block ?B
     (occurs :slot actor-pronoun [2]))
    (nowhere (follows [2] [1]))))
  (=>)
  (new parsing-hypothesis ?P2 extends ?P1)
   (in parsing-hypothesis ?P2
    (in-sentence (extension-of ?S))
     (replace-block-with (extension-of ?B)
      (occurs :slot actor ?ACTOR)))))
```

Whereas there's nothing new in the rule-based parsing *per se*, it serves as a demonstration of the system's flexibility. Furthermore, since the final results are narratives rather than parse trees, it also provides a rudimentary semantic parser.

Currently, the system can translate sentences with simple causal and physical relations it already has knowledge about. For example, sentences such as

```
Pink wanted the stick to be attached to the board, so he did it.
```

run through a disambiguating parsing Critic network consisting of *detect-goal, detect-temporal-relation, normalize-tense, same-sentence-actor-pronoun=¿disambiguate, goal-action-hypothesis=¿elaborate, relation-achieving-action=¿elaborate-observations*, and the following Narrative is extracted:

```
(desires pink (is-attached stick board))
 (sequential
   (observes pink (not (is-attached stick board)))
   (does pink (attaches pink stick board) [1])
   (observes pink (is-attached stick board) [2]))
 (causes [1] [2]))
```

The system cannot learn completely new relations from natural language yet, since the parser Critics won't know which kind of narrative predicate to translate them into, and their parsing hypotheses will be rejected. Utilizing statistical parsers to deduce the appropriate frame slots could help with the problem for simple relations, but proceeding beyond that would require significant research effort.

3.2 Uncertainties

Singh's EM-ONE is based on graphic pattern matching of binary predicates. The diversified evaluation systems provided by the assessment networks by themselves provide an implicit mechanism to handle logical contradictions and deciding functions for uncertainties. However, in more complicated situations, it is increasingly necessary to make educated guesses, if only to conserve resources.

Critics don't necessarily need all conditions to match in their antecedent sides. Some Critics are useful in certain situations even if not all of the normal preconditions were filled, such as shouting for help even though no-one else is seen around:

```
(defcritic reactive*in-trouble=>shout-for-help
 (necessarily
  (in-trouble ?ACTOR))
 (preferably
  (in conditions current-conditions
   (observes :actor ?ACTOR :thing ?OTHER-ACTOR)
   (different ?ACTOR ?OTHER-ACTOR)))
 (=>)
 (in conditions current-conditions
  (does :actor ?ACTOR
        :prop (shouts :actor ?ACTOR
                      :object "Help!"))))
```

Other common situations where uncertainty predicates can be used are where current readiness to perform a particular task has influenced the ability to eventually be successful; a Critic assisting in the use of a weapon to defend against an attacking enemy could deem 1) the existence of the weapon as a *necessary* component, 2) the weapon already being in your hand as a *preferable* component, and 3) the weapon being on your body as an *optional positive* component.

The system treats all Critics without an uncertainty predicate as all clauses needing to be matched. When they exist, the necessary components need to be matched, whereas the preferable and optional parts contribute to the weight score used in the evaluation of the hypothesis produced by the Critic. Currently, the weights are adjusted by a fixed amount according to the predicate used.

Experimentally, different degrees of necessity can be used to blur the line between strict binary truth value matching and probabilistic reasoning. Concretely, they can be used to influence the way the critic is used when evaluating hypotheses: by giving different weights, they can report a score for how the Critic is expected to hold in some particular situation, rather than just a binary evaluation. Furthermore, it would be possible to train these scores in different contexts, gaining an optimizing model across a wide range of commonsense domains - parts of Critics could have differing importances in different contexts.

3.3 Personal Modeling

The original EM-ONE has an omniscient viewpoint; the knowledge in the Fact database can be accessed by any of the simulated robots and doesn't 'belong' to anyone. Individualization is done by the **:actor** slots of the *desires, believes, observes, does, intends* predicates. This means that it is possible to write Critics that 'read the mind' of other robots.

In the improved version, this is prevented by having each actor in the simulation have their own individual knowledge bases and critic networks.

3.4 Learning Critic Networks

The existing scene-by-scene Metacritic networks can be regarded as hand-coded high level K-Lines [Minsky, 1980][Minsky, 1986]: in response to a particular type of a problem, a selection of mental resources is activated to deal with it. Based on the effects of the activation of the network, the reflective layer of Critics can modify network if it's found to have produced an inaccurate or unhelpful assessment of the situation.

Thus, a Critic Network is simply a collection of Critics, but encompasses a broader range of methods for identifying and dealing with problems. The language for specifying networks is the same as for Critics, so the difference is mainly a conceptual one: whereas Critics are general methods that pair with narratives to generate specific hypotheses, the networks work both to narrow down the number of Critics that need to be used, and make instruments out of Critics to match specific problems with specific methods.

Whereas we probably need some existing patterns to produce successful problem-solving behavior, we could try learning new ones. Equipped with an array of Critics, and possessing some goals, we can release an agent to a simulated environment, where it can to try and find Critic networks to achieve those goals. Feedback for problem solving with purely physical objectives can be acquired through directly observed world state changes in the physical world. Feedback for problem solving involving other intelligent agents could be achieved by involving human-controlled agents, or other AI agents.

The program saves a trace of which Critics have been utilized, using what resources, in attempts to solve a problem - this is what the reflective Critics rely on. By saving the traces of both the successful attempts and particularly disastrous ones, we learn K-lines appropriate for specific situations. These can be used both as Critc networks and as activity propagators to 'wake up' relevant resources in a performance-conscious system.

3.5 Self-Reporting

The Critic and symbol names are intended to be concise, so simply reading traces of Critic execution and generated and accepted hypotheses provides enough debugging information for the programmer to determine exactly why the program arrived at any particular decision. Furthermore, since the symbols used internally and for data entry are the same, inserting knowledge such as 'for breathing and breeding purposes, whale is a mammal, but for movement purposes, whale is a fish' is straightforward.

Eventually, though, we might get to a stage where it would be convenient if the system could ask a human direct questions to solve disambiguities, much as human children do. The capacity for questioning has not been implemented yet, but generating English from the internal representation can be done, even though the result is currently quite monotonous.

For example, using a Critic network with 6 conversion patterns, the narrative in Section 2.1 becomes *"This is a story about 'attaching-stick'. First, Pink observes that stick is not attached to board. Then, Pink attaches stick to board. This causes that, then, Pink observes that stick is attached to board."*

4 Discussion and Future Work

What kinds of cognitive behavior does a Critic-based system enable particularly well, and what is it bad at? Where is it better than expert systems, planners, neural networks, statistics?

The system described here is, at its core, a processor for a language which describes mental events. As with all languages, it only sets the framework, and what can be done with it depends on the creativity and insight of the authors using it. By making networks of Critics, the system can learn about successful special cases on its own; but it is hard to imagine the system independently producing any elementary Critics in the near future stages of development.

A clear shortcoming we have is dependence on cleanly structured and disambiguated symbolic knowledge of world events, databases of which are notoriously difficult to construct. The Critic-L based system alleviates the problem in two ways: first, predicates in the language can refer to non-symbolic processes; second, the mechanism itself can be used to construct more knowledge for its own consumption, achieving rudimentary bootstrapping.

The following are experimental areas for future development.

4.1 Imagination

An area which benefits from being able to express things symbolically is *imagination*. Using symbolic cutting and splicing, we can imagine conditions that are slightly or completely different from our current or remembered surroundings.

Using imagination as problem-solving tool is powerful, since we don't need to be constrained by typical predictions derivable from the current context. We can wish things into existence in our imaginary contexts, and if those things are helpful, use the imaginary contexts as potential subgoals, creating shortcuts in the search space. For example, if we need to cross a river, we can imagine a bridge or a raft, even if we don't see one, and proceed to find or make it.

The current system implements imagination in a roundabout way, by a collection of Critics that create, modify and access hypothetical conditions. However, cues extracted from commonsense narratives are necessary for guessing which kinds of imagined scenes could be useful. For the purposes of crossing it, imagining a river drying up is hardly helpful unless one has access to a dam upstream.

We are planning to expand the selection of creative imagination Critics by including cues involving object properties other than their stated relations in a narrative, such as physical shape, size and weight.

4.2 Interfacing to Different AI paradigms

Singh's original EM-ONE has an interface to the Roboverse world simulator where relevant properties of the physical world state are heuristically translated into world describing predicates that the Critics can understand. Our program detects problems by graph-based pattern search and generation, but as long as it's possible to construct a suitable translator function, the system can also make use of AI paradigms relying on different kinds of data.

Currently, work is being done to incorporate knowledge acquired through machine learning. The following is a hypothetical Critic for detecting that a board game, such as Go [Silver *et al.*, 2016], is being played, and generates a neural network and a move-providing encapsulated construct for helping with the game.

In this example, the *two-actor-repeating-pattern* is a pattern matching process tracking similar moves being repeated with identical objects, and *game-pattern* by detecting end states where repetition ends and wins are associated with the winner smiling.

```
(defcritic meta*detect-game=>generate-nn
 (two-actor-repeating-pattern ?P ?ACTOR1 ?ACTOR2)
 (game-pattern ?P ?GAME)
 (game-winstates ?GAME ?W)
 (=>)
 (learn-game-neural-network ?P ?W ?MOVE-GENARATOR)
 (assert-block
  (in narratives (generate-narrative :basic)
   (sequential
    (desires ?SELF (win-game ?SELF ?GAME))
    (extract-endstates ?MOVE-GENERATOR ?END)
    (repeating-pattern
     (until ?END)
     (does ?SELF (game-move ?SELF ?MOVE-GENERATOR))
     (does ?OTHER (game-move ?OTHER)) [1])
    (observes ?SELF (win-game ?SELF ?GAME)) [2])
   (causes [1] [2]))))
```

With access to sufficiently powerful APIs, it should be possible to construct complex predicates such as the *learn-game-neural-network*. In this example, we end up with a MOVE-GENERATOR that is a black box as far as the rest of the system is concerned.

5 Conclusion

In this paper, we have reviewed the main components of our improved version of EM-ONE with an emphasis on their modularity and suitability for serving as abstractions for different kinds of knowledge. Singh's original Critics and Narratives are coded by hand, but he briefly discusses methods for acquiring them through learning; this paper introduces some of the preconditions necessary for acquisition from messier, real-world-like environments.

The Critic-based model described here is a self-reporting and self-inspecting knowledge processing mechanism, where the core components are hand-crafted, but they can modify each other based on experience, allowing for learning.

The system is a work in progress, and it can currently build physical objects such as tables and arches, report on its reasoning and communicate intent. With an expanded, large enough knowledge base, it could conceivably perform the 'constructing' part of Ortiz's Construction challenge [Ortiz, 2016] in a simulated environment. Reading of diagrams, actual speech recognition and real-world motor control would need to be performed by additional components.

Besides modularity, a particular strength of our system is its readability. Unlike the data representations used by neural networks, the Narrative-L and Critic-L are conceptually similar enough to natural language that they can be easily read when debugging, and authored without too much difficulty. We expect the readability to be an asset in the future, since being able to elucidate the reasons behind your actions, and discuss alternative courses of action in different conditions is one of the hallmarks of human intelligence, but is sadly lacking in most modern AI work.

6 Acknowledgements

Marvin Minsky provided valuable feedback for developing the ideas expressed in this paper.

References

[Havasi *et al.*, 2007] Catherine Havasi, Robert Speer, and Jason Alonso. Conceptnet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent advances in natural language processing*, pages 27–29. Citeseer, 2007.

[Lenat, 1995] Douglas B Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[Minsky *et al.*, 2004] Marvin Minsky, Push Singh, and Aaron Sloman. The st. thomas common sense symposium: designing architectures for human-level intelligence. *AI Magazine*, 25(2):113–124, Summer 2004.

[Minsky, 1980] Marvin Minsky. K-lines, a theory of memory. *Cognitive Science*, 4:117–133, 1980.

[Minsky, 1986] Marvin Minsky. *The Society of Mind*. Simon and Schuster, New York, 1986.

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

[Morgan, 2013] Bo Morgan. *A Substrate for Accountable Layered Systems*. PhD thesis, MIT, 2013.

[Norvig, 1991] Peter Norvig. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann, 1991.

[Ortiz, 2016] Charles Ortiz. Why we need a physically embodied turing test and what it might look like. *AI Magazine*, 37:55–62, 2016.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[Singh, 2005] Push Singh. *EM-ONE: An Architecture for Reflective Commonsense Thinking*. PhD thesis, MIT, 2005.