Adaptive Algorithmic Hybrids for Human-Level Artificial Intelligence

Nicholas L. CASSIMATIS
Department of Cognitive Science
Rensselaer Polytechnic Institute

Abstract. The goal of this chapter is to outline the attention machine computational framework designed to make a significant advance towards creating systems with human-level intelligence (HLI). This work is based on the hypotheses that: 1. most characteristics of human-level intelligence are exhibited by some existing algorithm, but that no single algorithm exhibits all of the characteristics and that 2. creating a system that does exhibit HLI requires adaptive hybrids of these algorithms. Attention machines enable algorithms to be executed as sequences of attention fixations that are executed using the same set of common functions and thus can integrate algorithms from many different subfields of artificial intelligence. These hybrids enable the strengths of each algorithm to compensate for the weaknesses of others so that the total system exhibits more intelligence than had previously been possible.

Keywords. Human-level intelligence. Cognitive architectures.

1. Motivation

The goal of this chapter is to outline a computational framework that makes a significant, measurable advance towards creating systems with human-level intelligence (HLI). This work is based on the hypotheses that: 1. most characteristics of human-level intelligence are exhibited by some existing algorithm, but that no single algorithm exhibits all of the characteristics and that 2. creating a system that does exhibit HLI requires *adaptive hybrids* of these algorithms.

1.1.Why human-level intelligence

In this chapter, a system will be said to have *human-level intelligence* if it can solve the same kinds of problems and make the same kinds of inferences that humans can, even though it might not use mechanisms similar to those humans in the human brain. The modifier "human-level" is intended to differentiate such systems from artificial intelligence systems that excel in some relatively narrow realm, but do not exhibit the wide-ranging cognitive abilities that humans do. Although the goal is far off and there is no formal characterization of human cognitive ability, there are several reasons for adopting it:

1. Assuming humans are entirely composed of matter that can be simulated by computers, then if a human can do X, a (sufficiently powerful but still physically realizable) computer can do X also. Thus, while factoring million-bit numbers in a few seconds may be beyond the reach of any physically realizable computer,

activities such as having a human-language conversation or discovering a cure for a disease are not.

2. The applications of human-level intelligence would be tremendous.

3. Ideas behind a human-level artificial intelligence are likely to help cognitive scientists attempting to model human intelligence.

4. Decisions in designing intelligent systems often require tradeoffs, for example between soundness and completeness vs. quick computation. Since human beings are an existence proof that a system can be quite powerful without sound and complete planning and decision making, we know that a system that does not offer such guarantees would nevertheless be quite powerful. Thus, the human case can provide motivation or justification for making certain tradeoffs.

1.2.Problem of tradeoffs

There are several characteristics of AI systems we want. The problem is that in the case of human-level intelligence, most existing algorithms exhibit some characteristics at the expense of others. For example:

Generality vs. speed. Many search, Bayes network and logic-theorem proving algorithms are often (in the limit, at least) guaranteed to produce a correct answer. This makes them very general. However, for problems involving many state variables, these algorithms are often too slow. More "structured" algorithms such as case-based reasoning or script matching can produce quick results even on problems with enormous state spaces. However, these algorithms often have trouble when there is no good case or script that matches a new situation. They thus trade speed for generality.

Complex vs. robust behavior. "Traditional" planning algorithms are capable of constructing complex sequences of actions to achieve goals. However, in a system with noisy sensors in a changing world, plans are often invalidated in the time it takes to formulate them. Reactive systems (e.g.,[1];[2]) quickly react to an existing situation while often not creating or manipulating any model of the world. This, however, is a problem in situations that require complex plans about unseen, past and/or future parts of the world. Reactive systems therefore often trade flexibility for complexity.

These sorts of tradeoffs are not an obstacle in many domains. For example, problems that can be formulated with a modest and fixed number of state variables can often be successfully and quickly solved using SAT-based search algorithms. In this case, speed and generality are both possible.

However, many problems that humans can solve are so large and open-ended that tradeoffs in existing algorithms seem to become difficult to avoid. For example, a system that must read, summarize, make inferences from and answer questions about press reports on, for example, the biotech industry, faces several problems[1]:

Large state space/knowledge base. The amount of background knowledge that can be brought to bear in any particular story is enormous. Understanding and making inferences from a sentence such as "The discovery of the botulism vial in Sudan just before Thanksgiving has increased the value of companies targeting proteases enzymes" requires knowledge of biology, terrorism, drug discovery, stock prices and US Holidays. Any system that makes human-level inferences based on such text must therefore operate in a state space with an enormous number of state variables. This is

[1] The progress of text retrieval, summarization and question-answering systems in this domain do not mean this is a solved problem. These systems are still at best approximate human experts and are not nearly as capable as they are.

often true for many problems humans can solve. Such large state spaces make it very difficult or impossible to avoid tradeoffs between generality and speed.

Time and equality. State variables can change over time. There are on the order of a billion seconds in a human lifetime. This significantly exacerbates the problem of state-space size. A similar problem is caused by identity. We often cannot perceive, but must infer, that an object seen now is the same as an object that was seen earlier. The possibility of such identities can greatly increase a state space.

Open world. Even in the simplest scenarios, inferring the existence of previously unknown objects is routine. For example, someone seeing a ball on a table roll behind an occluding object and not emerge from behind it can infer that there must be something like a hole, obstacle or adhesive substance which halted the ball's motion. However, many algorithms assume that the objects that exist are specified and fixed before reasoning begins. This is the case for many search, Bayesian reasoning and logic-theorem proving algorithms. When this assumption does not hold, these algorithms often either makes them inefficient or inoperable. For example, it is difficult to stop searching for a proof if adding another object to the universe is an option an algorithm has before giving up.

Changing world and noisy sensors. New sensor readings or changes in the world often invalidate an algorithm's inferences or choices. One option is for algorithms to be rerun each time information changes. This places a severe time constraint on them since they must be ready to rerun as soon as new information comes in. Another option is for algorithms to be modified to take new information into account as they are being executed. This is a much harder problem given how most complex algorithms are implemented today.

1.3.Uniform and modular approaches to the tradeoff problem

There are two common ways to avoid making these tradeoffs. First, extend an existing computational method so that it does not make a trade. For example, dynamic Bayes networks the fact that state variable values can change over time. DBNs thus reduce the expressive power one must trade to achieve (approximately) probabilistically correct inference. There are many similar efforts in many subfields, but the work in this project is based on the hypothesis that the drawbacks of individual classes of algorithms cannot ever completely be removed and that at a minimum it is worth exploring other alternatives to algorithmic uniformity.

One such alternative is to create computational frameworks for combining modules based on different data structures and algorithms. One potential problem with these approaches is that the tightness of integration between algorithms and data structures may not be strong enough. For example, it is conceivable that grounding a literal in a logic theorem proving module might be accomplished by using a vision system, a neural network and/or database access. While the work in this project employs some modular integration, it is based on the hypothesis that modular integration alone will not be sufficient and that, somehow, every single step of many algorithms can and must be integrated with many other algorithms in order to achieve HLI.

Both approaches – extending a single computational method and modular architectures – have achieved significant success. It is difficult to decisively argue that the problems just mentioned cannot be overcome. Nevertheless, since these approaches are already being studied by so many, this chapter explores the possibility of another approach to resolving tradeoffs between different computational methods.

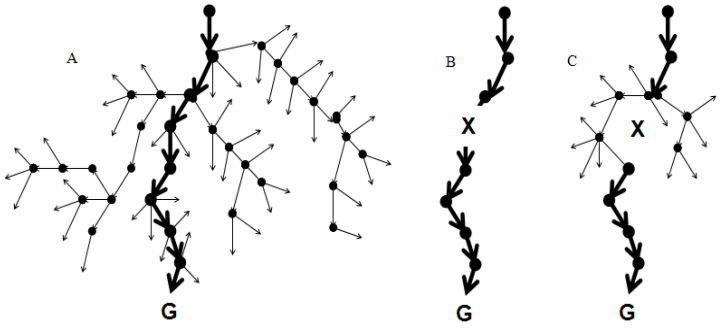

Figure 1. A hybrid (C) of systematic, general, flexible but slow search (A) with fast but rigid case-based reasoning (B).

1.4.Adaptive hybrids

The overall approach of this work is not only to create systems that involve many algorithms in modules, but to execute algorithms that exhibit the characteristics of multiple algorithms depending on the situation. An example of what this might look like is illustrated in Figure 1. Figure 1a depicts a search tree that finds a plan. The "bushiness" of the tree illustrates how search is slowed down by having to try many possible options. Figure 1b illustrates a case-based planner retrieving this plan when a similar situation arises. The "X" illustrates an imperfection in a plan (i.e., an obstacle to its successful application) and illustrates the potential rigidity of a pure case-retrieval approach. Figure 1c illustrates a hybrid of search and case-based reasoning that correctly solves the problem. Most of the problem is solved by case-application, but in the region where there is a problem with the case, search is used to solve the problem.

The goal is to create systems that execute such hybrids of algorithms and adapt the "mix" of the algorithms to changing characteristics of the situation. The intended result of this is the ability to create systems that exhibit more characteristics of HLI together in one system than has been possible so far. In addition to demonstrating the power of the approach on challenge problems in microworlds, we also intend to construct systems that measurably produce a significant advance in at least one important application domain.

2. Algorithms as different ways of exploring a multiverse

In order to motivate the design of a computational system that implements hybrid algorithms, it is helpful to conceive of these algorithms within a common formal framework. Developing such a framework would be key component of the work in this project. This section provides a preliminary sketch which motivates the *attention machines* presented in subsequent sections. Although much of the terminology here is borrowed from first-order logic and probability theory, this project is not an attempt to reduce all of AI to one or both. These are merely formalisms for *describing* the goals

and capabilities of agents and algorithms. Attention machines are intentionally designed to include methods not normally associated with logic or probability theory.

2.1.Multiverse

Intuitively, a *multiverse* is the set of all possible worlds. Each world includes a history of past, current and future states. Multiverses are specified in terms of propositions. $R(a_1, ..., a_n , w)$ states that relation R holds over the a_i in a possible world, w. Arguments and worlds are drawn from the set of *entities, E*, where W, a subset of E, is the set of possible worlds in the multiverse. A multiverse is a triple, (M, E, W), where M is the set of all true propositions. $R(a_1, ..., a_n)$ is said to be true in world, w, if $R(a_1, ..., a_n, w)$ is a subset of M. The set of worlds is a subset of the set of entities so that one can state declarative facts about worlds, for example, to say that one world is counterfactual relative to another. Worlds are related to, but different from situations in the situation calculus. Situations conflate a representation of time and possibility. There is no temporal order among worlds. Each world "contains" a full (version of a) history.

2.1.1.Regularities in a multiverse

Regularities that knowledge representation schemes capture can be reflected in a multiverse. For example, the logical formula, $(x) (R(x) \rightarrow S(x))$ can be modeled by a multiverse in which, for every world w, where $R(e)$ is true for some e, $S(e)$ is also true. Probabilistic regularities can be modeled by treating all the possible worlds as equiprobable. Thus, $P(R(x)) = |W_r|/|W|$, where W_r is the set of all worlds where $R(x)$ is true. Conditional probabilities can be similarly modeled[2].

2.1.2.Formulating the goals of algorithms in the multiverse

It is possible to characterize the goals of algorithms from different computational frameworks using multiverses. This will enable a characterization of their operation that will motivate a computational approach for executing hybrids of these algorithms. This chapter will choose simple versions of problems from multiple subfields of artificial intelligence to illustrate this point.

Search. There are so many search algorithms that it is difficult to give a single characterization of what they are all designed to do. Many search algorithms can be thought of as methods for finding a state of affairs that satisfy some constraint. This is the purpose of most SAT-based search algorithms (see ([3]) for a review) and many other search problems can reduce to searches for models that satisfy constraints (e.g., STRIPS planning ([4])). In terms of the multiverse, these search algorithms try to find a possible world where these constraints are satisfied. In many typical single-agent planning contexts, for example, search aims to find a possible world in which a goal constraint is satisfied at a time in the future such that each difference between that future state and the current state results from an action of the agent, directly or indirectly.

Graphical models. Graphical models are used to capture conditional probabilities among state variables and to compute the probability distribution over a set of variables given observed values of those variables. Bayesian networks ([5]) are perhaps the best-known class of graphical models. A node in a network representing state variable X set

[2] This treatment of probabilistic relationships, clearly applies only when the set of all possible worlds is finite, though extending this approach to multiverses with infinitely many possible worlds should be straightforward.

to value *a* can be represented with the proposition *Value(X,a)*. Of course, there is no reason that a more expressive scheme cannot be adopted for specific applications (e.g., using *Color(dog,black)* to represent that the state variable corresponding to a dog's color state variable being set to 'black'). The prior and conditional probabilities in a Bayes Network (which correspond to edges in the network) can be captured by treating possible worlds in a multiverse as equiprobable, as outlined in the previous subsection. In multiverse terminology, the aim of a Bayes Network propagation algorithm is to find, for each state variable X and value v, the proportion of possible worlds where $X=V$. Thus, when one of these algorithms determines that $P(X=a) = p$, it is estimating or computing that *Value(X,a)* is true in $p*N$ possible worlds, where N is the total number of possible worlds.

Logic programming. Logic programming systems enable knowledge to be declared in a logical formalism and queries about what that knowledge entails to be automatically answered, often by searching for proof of a theorem. The clauses and literals which make up logic programs are straightforwardly mapped onto a multiverse. The last subsection showed how a multiverse could capture material implication (clauses)[3]. Thus, the goal of a logic programming algorithm determining whether a closed literal L is true is to somehow (e.g., by searching for a proof or failing to find a model where L is false) show that that proposition P_L representing L is true in all possible worlds in the multiverse.

2.2.The execution of AI algorithms as paths through the multiverse

It is possible to characterize, not only the goals, but the operation of algorithms from multiple subfields of AI within the same multiverse. This motivates a computational architecture for implementing hybrids of these algorithms. The key idea, which we call the **common operation principle**, is to recognize that these algorithms can be decomposed into the same set of "common operations" which each involve exploring some part of the multiverse. A provisional set of common operations developed in preliminary work include:

Forward inference. Given the truth values of a set of propositions, *P,* produce a set of propositions, *Q*, entailed or made more likely by the those truth values.

Subgoaling. Given the goal of determining whether a proposition *Q* is true, produce a set of propositions, *P*, whose truth values would make *Q* more or less likely.

Grounding. Given a proposition, *P*, with open variables, return a set of true closed propositions that are identical to *P* under some assignment.

Identity and similarity matching. Given an entity, *e*, and a set of true and false propositions about *e*, return a set of entities *E* that are similar or (likely to be) identical to *e*.

Exploring possible worlds. Given a world, *w*, and a set of proposition-truth value pairs, return a world w_1 such that those propositions have those truth values and where otherwise everything that is true or false in *w* is so in w_1.

When an algorithm performs one of these operations on a proposition, we say that it *attends to* or *fixes its attention* on this proposition. We will also call this event an *attention fixation.* A key insight motivating this chapter is that **AI algorithms from**

[3] Literals with functions, e.g., which Prolog permits and Datalog does not, can be flattened to correspond to literals in a multiverse.

different subfields based on different computational formalisms can all be conceived of as strategies guiding attention through propositions in the multiverse.

Search. We illustrate how to frame search in the multiverse context using GSAT ([6]) because it is relatively simple, though extending this approach to other algorithms is straightforward.

Goal: Find a possible world where constraint *C* is true.
For MAX-TRIES:

- Start with a set of propositions, *P*, which can be true or false.
- For MAX-FLIPS
- Choose a world, *w*, by choosing a random subset of *P* to assign as true.
- If *C* is satisfied in *w*, then return *w*. *Forward inference.*
- Choose the proposition in *P* whose truth value changing will lead to the greatest decrease in the number of clauses in *C* being unsatisfied. *Subgoaling*.

In other words, GSAT, like search algorithms generally, explores the possible worlds in a multiverse. At every step, it chooses which world to explore by applying *forward inference* to the currently explored world to see if it is true (i.e., given truth values for propositions in a world, infer whether *C* holds in that world) and subgoaling (i.e., given the goal of making *C* true, find the propositions in the current world that will bring C closer to being true). Thus, GSAT can be characterized as a sequence of possible world exploration, forward inference and subgoaling in the multiverse.

Rejection sampling. Rejection sampling, an algorithm for probabilistic inference, can be straightforwardly recast as a method for exploring worlds in a multiverse in the following manner (recall that every node, *X*, in a network corresponds to an open proposition *Value(X,?value)*):

For *N* worlds, $w_1 \ldots w_n$:

- Every edge node, *E*, with an unknown value corresponding to proposition $Value(E,?value,w_i)$ grounds its value by sampling from the prior probability distribution corresponding to that edge. *Grounding.*
- Repeat until every state variable in the network has been set:
- For each interior node, *I*, whose input variables have been set:
- Given the input nodes and the conditional probability distribution associated with the edges leading into a node, sample a value, *V*, for *I*, yielding the proposition, $Value(I,V,w_i)$. *Forward Inference*.
- If the sampled value conflicts with an observed value stop exploring this world.

Estimate $P(X=v) \approx |\{w : Value(X,v,w)\}|/N$, where *N* is the number of possible worlds not thrown out during stochastic simulation.

Logic theorem proving. Many logic programming systems are based on algorithms such as SLD-resolution that repeatedly set subgoals and ground open literals. These algorithms can be straightforwardly formulated in terms of the common operations described above ([7]).

- *Case-based reasoning.* Consider the use of case-based reasoning to find a plan for dealing with a novel situation.
- Goal: Find an action, *a*, that will bring about effect, *e*, i.e., *Cause(a,e)*.

- Find an event in the past, *e′*, and action *a′*, in the past such that a′ caused e′ (*cause(a′,e′)*), *e′* is similar to the currently desired effect, *e*, (*Similar(e,e′)*) and such that *Similar(e,e′)*. *Identity match.*
- In the world, w_m, where *Cause(a,e)* and *Similar(a,a′)* (i.e., the world where the action taken to cause *e* is similar to the remembered action, *a′*). *Explore possible world.*
- For each of the roles or slots (r_1' … r_n') of *a*, find entities *(r_1… r_n)* such that *Similar(r_i', r_i,w_m)*. *Identity match.* (For example, if *a′* is the action of pounding a nail with a hammer, two roles are the hammer, *a*, and the nail, *n*. If the current goal is to insert a tent state into the ground, and the case-based reasoner decides to drive the stake into the ground with a rock, this is expressed by *Similar(hammer,rock,w_m)* and *Similar(nail,stake,w_m)*.)

Thus, a simple case-based reasoner can be constructed out of a combination of the similarity matching and possible world exploration common operations.

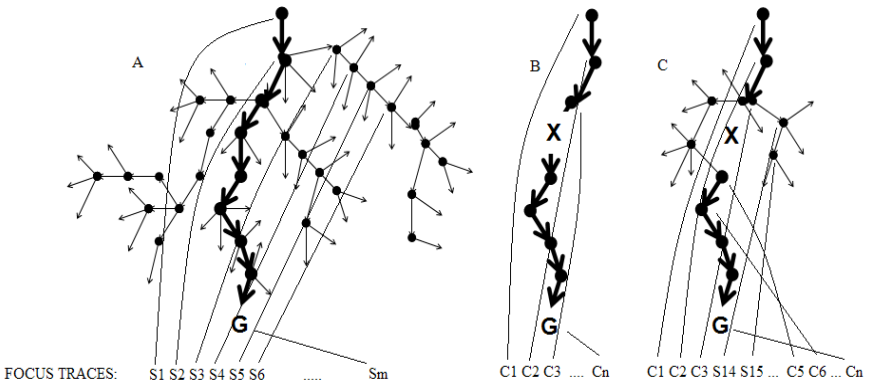

Figure 2. Algorithms and their focus traces. The focus trace (C) for the hybrid of case-based reasoning and search is a combination of the focus traces for search (A) and case-based reasoning (B).

2.3.Hybrid algorithms through hybrid focus traces

Each of these algorithms attends to (i.e. performs a common operation on) one proposition at a time. The sequence of attention fixations an algorithm makes when it executes is called its *focus trace*. Focus traces thus provide a uniform way of characterizing the execution of algorithms from different computational methods. This is a direct result of the common operation principle.

Focus traces also motivate an approach to executing hybrids of algorithms. One way to think about a hybrid of two algorithms is to think of hybrids of their focus traces. That is, an algorithm, H is a hybrid of algorithms A1 and A2 if H's focus trace includes fixations from both A1 and A2. For example, Figure 2 illustrates the hybrid execution of case-based reasoning and search. The focus trace (2c) for the hybrid of

case-based reasoning and search is a combination of the focus traces for search (2a) and case-based reasoning (2b).

Thus, if we can create a computational architecture that selects common operations from multiple algorithms, it would lay down a focus trace that would be a hybrid of those algorithms' focus traces. The next section sketches the *Attention Machine* architecture for executing hybrids in this manner.

3. Attention machines

Attention machines are systems that execute algorithms by sequences of common operations. Each common operation is implemented as an attention fixation. Attention machines enable hybrids of algorithms to be executed by interleaving sequences of attention fixations.

Formally, an attention machine (AM) is an ordered pair (S, FM), where S is a set of modules called *specialists* and FM is a *focus manager*.

The reason for including multiple specialists in an AM is that each common operation can be implemented using multiple data structures and algorithms. We call this the *multiple implementation principle.* This principle enables modules to compensate for each other's weaknesses. For example, neural networks are often better at making forward inferences about object categories than rule matchers, while rule matchers are often better at making forward inferences involving causal change than neural networks. Thus, depending on the situation a particular common operation may be better performed using one computational method over another. Having several computational methods (each encapsulated inside specialists) implement each common operation increases the chances that a common operation will be successfully performed. What happens when two specialists conflict in performing a common operation, e.g., when they each take a different stance on a proposition will be addressed later in this section.

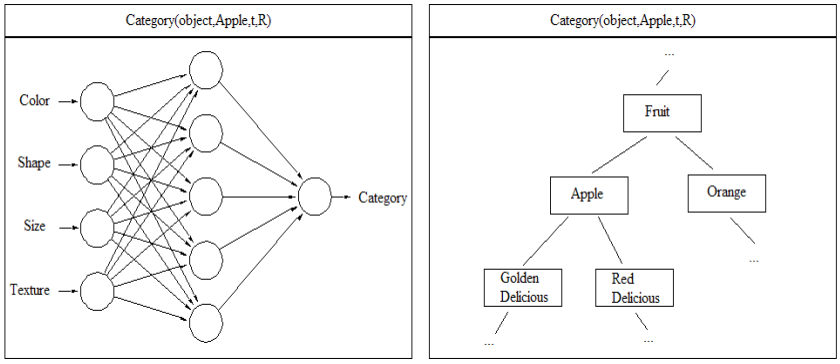

Figure 3. Two specialists using different internal representations but able to share information using a propositional interlingua. A classification specialist (left) stores information about categories in the connections of a neural network. An ontology specialist (right) stores information about categories in a graph. Both can translate this information from and into the same interlingua.

3.1.Sharing information with a propositional interlingua

If specialists use different data structures from other specialists, how can they share information? In AMs, this is accomplished through a specialist-neutral propositional interlingua that all specialists must be able to translate into their own. Working out the details of this interlingua is one of the objectives of this project, however preliminary results and other work suggest that a simple first-order propositional language might suffice. The reason for this is that the interlingua is used purely for communication, not inference. Note that an interlingua of FOL propositions does not commit the specialists or any other part of the project to logical techniques. Figure 3 illustrates two specialists using very different internal representations about the same object, but translating it the same interlingua.

3.2.Focus of attention

When do specialists share information? For several computational reasons, as well as an analogy with human visual attention ([8];[9];[10]), specialists in AMs all focus on and share information about a single proposition at a time. This minimizes the chance of a specialist making an inference based on an assumption another specialist knows is false. It also increases the chance that an inference by one specialist will be incorporated into other specialists' inference as soon as possible.

To share information with each other, the specialists implement the following functions:

- `ForwardInferences(P,TV)`. Given a new proposition `P` and its truth value, return a set of truth values for propositions which can now be inferred.
- `Subgoals(P)`. Return a set of propositions whose truth value would help determine `P`'s truth value.
- `Ground(P)`. Return a set of propositions which are a grounding of `P`.
- `SimilarityMatches(P)`. Return a set of propositions representing similarities supported by `P`. For example, learning that *x* is red (*Red(x)*) might lead to the inference that *x* is similar to *y* (*Similar(x,y)*).

3.3.Selecting attention

The best way to implement a focus manager is a topic this project aims to explore. However, a very simple scheme based on a "focus queue" will help illustrate this approach. Preliminary work demonstrates that even such a simple method can generate important results. We will assume, only for now, that the focus manager is a queue of propositions and that at every time step the first propositions on the queue is chosen. Specialists help guide attention by transforming the queue with the following procedure:

- `PrioritizeFocus(Q)`. Returns a queue that augments a version of Q modified to reflect the propositions the specialist wants to focus on.

At every time step in attention machines, all the specialists focus on the same proposition at the same time. Specifically, for an attention machine with focus manager, `FQ`, at every time step:

- The focus manager chooses a proposition, `focus-prop`, to focus on (in this simple case) by taking the first element of `FQ`.

- For every specialist, `S1`. ("All the specialists learn of each other's opinion on the focus of attention.")
- For every specialist, `S2`:
- `S1.Store(focus-prop, OpinionOn(focus-prop,S2))`.
- For each specialist, `S`, request propositions to be focused on:
- `FQ = S.PrioritizeFocus(FQ)`.

Thus, at every time step, specialists focus on a single proposition and execute their common operations on that proposition. In other words, the flow of computation in AMs is guided entirely by the choice of focal proposition.

Because the execution of an algorithm can be conceived of as a sequence of attention fixations (during which common operations are executed on the focused proposition), *how attention is selected determines which algorithm is executed.*

Let us now see how simple attention selection strategies can lead to focus traces that correspond to the execution of some key AI algorithms from different subfields. Simple versions of search, rejection sampling and case-based reasoning can be implemented with the following simple attention control strategies, embodied in the PrioritizeFocus procedure of a specialist, S.

- *Search*. When none of the specialists have any opinion on proposition, `R(x,y,w)`, or when those opinions conflict:
  - `PrioritizeFocus(FQ)` returns `addFront(R(x,y,w1),addFront(R(x,y,w0),FQ))`, where `addFront(X,Q)`returns a queue that is identical to `Q` with `X` added to the front, `w0` is the possible world where `R` does not hold over `x` and `y` and `w1` is the world where it does.
- *Rejection sampling*. When `S` believes that `R(x,y)` is *p/q* times more likely (for example by keeping track of the relative number of worlds in which `R(x,y)` is true):
  - `PrioritizeFocus(FQ)` returns `addFrontN(R(x,y,w1), addFrontN(R(x,y,w2), FQ, q), p)`, where `addFrontN(X,Q,N)` returns a queue identical to `Q` with N copies of `X` attached to the front of it and where w1 and w2 are as above. (In English, "focus on the world where `R(x,y,w1)` is true *p*/*q* times more often than the world where it is false.")
- *Truth maintenance*. When `S` changes its opinion on a proposition `P`:
  - `PrioritizeFocus(FQ)` returns `addFront(P,FQ)`. ("If you change your opinion on `P`, have all the specialists focus on `P` so they know your new opinion and can revise inferences they made based on that opinion on `P`.")
- *Case-based reasoning*. When having proposition `P` be true is a goal:
  - `PrioritizeFocus(FQ)` returns `A,R1 … Rn,` where
    - `S` has retrieved a proposition `A′` representing an action that achieved `P′` such that `Similar(A′,A)` and `Similar(P,P′)` and
    - `ri` are propositions of the form `Similar(ri,ri′,Wr)`, where `ri′` are participants in `A′` and `A` are participants in `A`. ("If `P` is a goal, retrieve actions that have achieved similar goals in

the past and find analogues for those participants in the present.")

This illustrates how the algorithms from different subfields can be implemented using the same basic common operations.

3.4.Resolving conflicts among specialists and inference strategies

How do AMs choose from among the several focus strategies described in the last section? In practice, this problem is less severe than it seems at first glance. A recurrent theme among the algorithms used to illustrate the common operation principle is that they choose which common operation to execute in response to *metacognitive problems:*

1. Search algorithms choose possible worlds by assuming the truth of propositions that are unknown. If the truth value of a proposition is known, it is considered fixed and worlds with that proposition having the opposite value are not explored. In the case of constraint-based search algorithms the metacogntive problem is ignorance, the truth value of the proposition is not known. In the case of planning-based search, the ignorance is often based on conflict, since there is more than one possible next action to take or explore.
2. Stochastic simulation algorithms also choose worlds to explore based on unknown values of a state variable, but the level of ignorance is somewhat reduced since a probability distribution for the state variable is known. This leads to a somewhat different exploration strategy exploring worlds with more likely truth values more often than those with less likely truth values.
3. Resolution-based logic theorem proving algorithms are also driven by ignorance. If the truth value of a literal was known, then an algorithm could simply retrieve it and there would be no need to search for a proof.
4. Case-based reasoning algorithms are also often driven by ignorance and differ from search and simulation algorithms by how they react to the ignorance. Instead of exploring possible worlds where different actions are taken, they retrieve similar solutions to similar problems and try to adapt them.

Thus, case-based reasoning, search and stochastic simulation are different ways of addressing three different kinds of *metacognitive problems*. This is called the *cognitive self-regulation principle*. Stochastic simulation deals with metacognitive problems where there is some information about the likelihood of different solutions to it working. Case-based reasoning deals with metacognitive problems where there are past solutions to similar problems. Search deals with the case where very little is known about the specific metacognitive problem except the basic knowledge about the domain that makes it possible to search for a solution.

Thus, the different algorithms deal with different kinds of metacognitive problems and will therefore conflict with each other rarely. In the current scheme, such rare conflicts are resolved by the order in which specialists modify the focus queue. One specialist can modify the queue and there is nothing to prevent the next specialist which modifies it to undo the modifications of the first specialist. In preliminary work, this has not yet caused any problems, though as implemented AMs and the problems they address become more complex, better methods of managing focus will be needed.

3.5.Extending the focus manager

The above discussion demonstrates how AMs with even a very simple focus management scheme (ordering propositions in a queue) can enable extensive hybridization. Queues, however, are almost certainly not a viable mechanisms for achieving the long-term goals of this project. Thus, part of every phrase of this research will be to explore ways of guiding the attention of specialists. In addition to focus management schemes that are motivated by a formal or empirical analysis of particular problems or domains, we will study how focus management can be learned.

One idea is to use state-action-reward reinforcement learning algorithms to automatically generate focus management schemes. The actions are the choice of attention fixation. The state will be both the state of the environment and the goals and subgoals of the AM. The reinforcement signal will be a combination of measures such as how many propositions are in doubt and how many goals are (close to being) achieved. Since in this framework the choice of attention fixation amounts to the choice of algorithm to execute, this approach can show how different kinds of AI algorithms are ways of optimizing cognitive effectiveness on different kinds of problems.

3.6.Summary: Two ways of achieving hybrids in AMs

We have described two ways to make hybrids of algorithms in AMs. The first is to execute a sequence of attention fixations, i.e., a focus trace, that is the combination of focus traces from individual algorithms. The key insight that makes this possible is that many algorithms from very different branches of AI, including case-based reasoning, logic-theorem proving and search can be implemented through sequences of common operations. Figure 2 illustrated this kind of hybridization.

The second way to achieve hybridization is by enabling modules based on many different computational methods to execute common operations. Every algorithm that is executed as a sequence of attention fixations is integrated with each of the algorithms inside of the specialist.

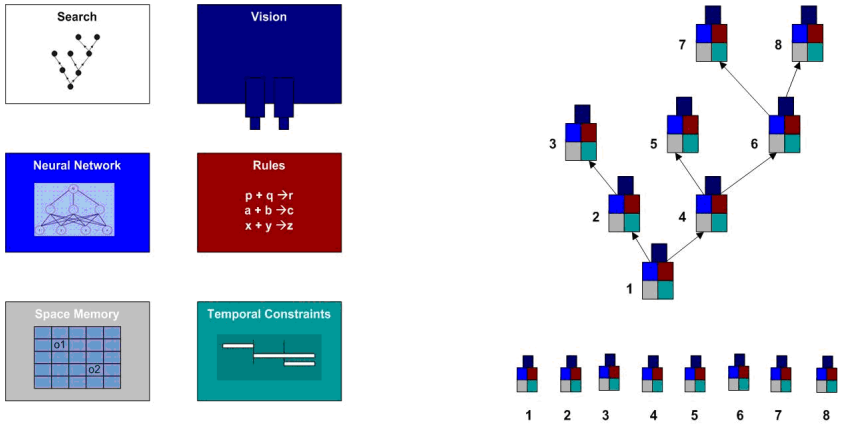

Figure 4. Purely modular integration compared to an attention machine hybrid.

Figure 4 illustrates this kind of integration. On the left, search is implemented in a purely modular system where it is isolated from other computational mechanisms. On the right, search is implemented as a sequence of attention fixations that each involves *all* the modules. Thus, every data structure and algorithm inside the modules is integrated with every step of search.

4. Results so far

Preliminary results with AMs are promising. They demonstrate that AMs can achieve quantifiable improvements over non-hybrid approaches and that they can enable new functionality in real-world systems. The most extensive AM implemented yet is a physical reasoning system for (real and simulated) robots. This section describes this system in relatively more detail as an illustration of how to implement and evaluate AMs. Some other preliminary results are also briefly presented.

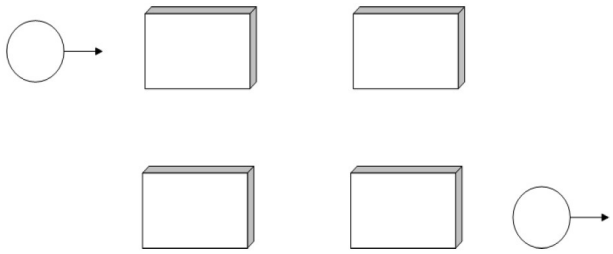

Figure 5. A physical reasoning problem. Is the object that rolls behind the first occluder the same as the object that rolled out of the second.

4.1.Physical reasoning

Figure 5 illustrates a (deceptively) simple physical reasoning problem. An object passes behind an occluder; a similar object emerges from another occluder. Are they the same object? The problem of making such inferences for arbitrary configurations and behaviors of objects includes the computational challenges mentioned in the first section. There is *incomplete information*; the state of the world in some locations is hidden from the view of the robot. It is an *open world*; there may be objects hidden behind occluders that are not perceived but which can still affect the outcome of events. The "size" of the problem is very large; even if we idealize space and time into a 100x100x100 grid with 100 possible temporal intervals, there are 10^8 possible place-time pairs that might or might not have an object in them. It is *a dynamic environment*: since objects can move independently of the robot, inferences and plans can be invalidated before they are even completely formulated.

This seemingly simple problem thus contains many of the elements which make AI difficult. It therefore causes difficulties for many existing approaches.

SAT search. SAT search algorithms must represent this domain using a set of propositional constraints. The constraint that objects trace spatiotemporally continuous

paths, for example, can be expressed with the following first-order conditional: *Location(o,p1,t1) ^ Location(o,p2,t2) ^ SucceedingTimeStep(t1,t2) → Adjacent(p1,p2)*. For O objects, an NxNxN grid and T time steps, the number of propositional constraints this compiles into is $N^6 \; x \; T \; x \; O$. In a 100x100x100 grid, with 10 objects and 100 temporal intervals, this is a trillion propositional constraints.

Bayesian Networks. Bayesian networks are also propositional and generally have similar problems as SAT as the number of state variables increase.

Logic. Identity (i.e., equality) is a challenge for logic programming systems. For example, for each two-place predicate, a clause such as the following is required: *x1 = x2 ^ y1 = y2 ^ R(x1,y1) → R(x2,y2)*. A backward-chaining theorem prover operating in a world with 100 objects would have 10,000 possible resolvants to check for each equality literal in a clause such as this.

Also, the closed-world assumption which logical, Bayesian and SAT algorithms typically make does not hold in this domain requiring substantial modifications to all of them. (Though, see (Milch, 2005) for some resent work in open-world probabilistic inference).

4.1.1.How S6 deals with computational challenges to HLI

The following briefly describes several elements of the design of an AM called S6 for conducting physical reasoning.

Open world, big state space. Graphical models, SAT solvers and many kinds of planners is that they instantiate every state variable. In S6, state variables (e.g., representing objects, locations, colors, etc.) are not explicitly represented by specialists until they are focused on. Also, inference is not conducted through a brute-force search or Monte Carlo simulation of the state space. Inference is instead lazy. When a specialist can make a forward inference from a proposition that is focused on it makes it.

Identity. S6 only considers identities between objects that are suggested by a neural network. The network takes as input the properties of an object and outputs potential objects that might be identical to the input object. This greatly reduces the length of inference.

Reactivity. Since it is implemented in an AM, every step of search in S6 is conducted with an attention fixation that involves all of the specialist. This includes specialists that encapsulate sensors. Thus, every step of inference (i.e., every attention fixation) can be informed by new information from a change world. Further, if a perceptual specialist receives information that contradicts something a specialist previously believed, it will request that this new information is focused on. When the new information is focused on, specialists will redo forward inference, retract invalidated inferences and make new inferences based on the information. Thus, having every step of an algorithm implemented as focus of attention of specialists that include perceptual abilities enables inference to react and adjust to new or changed information the moment it is sensed.

4.1.2.Preliminary evaluation of physical reasoner

Several preliminary results attained with S6 suggest hybrid algorithms implemented in AMs can achieve significant improvements over algorithms executing individually, especially on problems with an open world, time, identity, a dynamic environment and noisy sensors.

The first result is that hybrid algorithms in S6 exists and makes inferences in domains where individual algorithms have trouble. For example, a subset of the physical reasoning problem S6 solves was coded in the Discrete Event Calculus ([11]) which uses SAT solvers to solve problems in the event calculus. For an 8x8x8 world with 8 time steps, the computer's available memory (450MB) was exhausted. S6 solved these problems easily. These comparisons need to be conducted with probabilistic inference systems and logic-theorem provers.

These results, of course, come at the cost of soundness and completeness. We suspect that this is a necessary tradeoff and therefore will compare inferences S6 makes against human performance in addition to normative standards. In preliminary work, S6 has already been attained success on many problems in the human physical reasoning literature (e.g.,[12])

S6 demonstrates that hybridization can lead to significant increases in efficiency. For example, as describe above, S6 uses a neural network specialist to suggest equalities. Only these equalities are considered by S6. When this neural network is disabled and search is run alone, S6, becomes extremely slow. The effect has been so dramatic (i.e., several orders of magnitude), that we have not carefully measured it yet, although this is a priority for the near future.

Finally, S6 has been used to design a robotic framework ([13]) for addressing the tension between the apparent rigidity and inflexibility of the sense-plan-act loop of traditional planning algorithms and the autonomy and reactivity required of real-world robots. This framework used specialists to implement reactive subsystems and used a simple attention management scheme to marshal these reactive components to produce deliberate planning and reasoning. The framework was used to create robots that executed traditional planning and reasoning algorithms (e.g., search, rule-matching, means-ends analysis) and could, the minute it was perceived, use changed or updated information about the world to revise past inferences.

4.2.Other results

Cassimatis (2003) described a system called, Polylog, that implements a logic programming system based on nonlogical data structures and algorithms. Polylog enables specialists based on any data structure or algorithm (for example, relational databases, numerical methods or neural networks) to be integrated in one system and return sound and complete answers to queries. Polylog implemented theorem proving using a focus queue and an attention controls strategy that iteratively made subgoals and grounded propositions. This work suggests that in some cases the tensions between the formal completeness of logic programming and the computational efficiency of specialized data structures and algorithms could be resolved using AMs.

None of this previous work implements the bulk of the ideas described earlier. The attention management schemes were preliminary and simple and many different computational methods were neglected (e.g, graphical models and POMDPs.) Nevertheless, this work demonstrates that the approach of creating hybrid algorithms by implementing them using common operations implemented by modules based on different representations enables a greater degree of robustness.

5. Conclusions

The goal of the work reported in this chapter has been to advance the ability of artificial systems to exhibit human-level intelligence by providing a framework for implementing adaptive algorithmic hybrids. By enabling algorithms to be executed as sequences of attention fixations that are executed using the same set of common functions, it is possible to integrate algorithms from many different subfields of artificial intelligence. This lets the strengths of each algorithm to compensate for the weaknesses of others so that the total system exhibits more intelligence than had previously been possible.

References

[1] P. Agre and D.Chapman, "What are plans for?," Journal for Robotics and Autonomous Systems, vol. 6, pp. 17-34, 1990.

[2] R. A. Brooks, "Intelligence without representation," Artificial Intelligence, vol. 47, pp. 139-159, 1991.

[3] H. H. Hoos and T. Stützle, "SATLIB: An Online Resource for Research on SAT," in SAT 2000, I.P.Gent, H.v.Maaren, and T.Walsh, Eds.: IOS Press, 2002, pp. 283-292.

[4] H. Kautz and B. Selman, "Unifying SAT-based and Graph-based Planning," presented at IJCAI-99, 1999.

[5] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufmann, 1988.

[6] J. Gu, "Efficient Local Search for Very Large-Scale Satisfiability Problems," SIGART Bulletin, vol. 3, pp. 8-12, 1992.

[7] N. L. Cassimatis, "A Framework for Answering Queries using Multiple Representation and Inference Technique," presented at 10th International Workshop on Knowledge Representation meets Databases., 2003.

[8] A. M. Treisman and G. Gelade, "A feature integration theory of attention," Cognitive Psychology, vol. 12, pp. 97-136, 1980.

[9] B. J. Baars, A Cognitive Theory of Consciousness. Cambridge: Cambridge University Press, 1988.

[10] J. R. Stroop, "Studies of interference in serial verbal reactions," Journal of Experimental Psychology: General, pp. 622-643, 1935.

[11] E. T. Mueller and G. Sutcliffe, "Reasoning in the event calculus using first-order automated theorem proving," presented at Eighteenth International Florida Artificial Intelligence Research Society Conference, 2005.

[12] E. S. Spelke, "Principles of Object Perception," Cognitive Science, vol. 14, pp. 29-56, 1990.

[13] N. L. Cassimatis, J. G. Trafton, M. Bugajska, and A. C. Schultz, "Integrating Cognition, Perception, and Action through Mental Simulation in Robots," Robotics and Autonomous Systems, vol. 49, pp. 13-23, 2004.