Logic, Self-awareness and Self-improvement: the Metacognitive Loop and the Problem of Brittleness

MICHAEL L. ANDERSON, *Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA.*
E-mail: anderson@cs.umd.edu

DONALD R. PERLIS, *Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA.*
E-mail: perlis@cs.umd.edu

Abstract

This essay describes a general approach to building perturbation-tolerant autonomous systems, based on the conviction that artificial agents should be able to *notice* when something is amiss, *assess* the anomaly, and *guide* a solution into place. This basic strategy of self-guided learning is termed the *metacognitive loop*; it involves system monitoring, reasoning about, and, when necessary, altering its own decision-making components. This paper (a) argues that equipping agents with a metacognitive loop can help to overcome the brittleness problem, (b) details the metacognitive loop and its relation to ongoing work on time-sensitive commonsense reasoning, (c) describes specific, implemented systems whose perturbation tolerance was improved by adding a metacognitive loop, and (d) outlines both short-term and long-term research agendas.

Keywords: Metareasoning, time, non-monotonic reasoning, active logic, brittleness, autonomous agents.

1 Introduction and background

Brittleness is arguably the single most important problem in AI, and perhaps in (computer) systems overall: a system designed for specific tasks fails utterly when faced with unanticipated perturbations that take it even slightly outside its task specifications. Yet humans perform admirably under such perturbations, easily adjusting to most minor changes as well as to many major ones.

We define a perturbation as any change, whether in the world or in the system itself, that impacts performance. Performance is meant to be construed broadly to encompass such things as reasoning efficiency and throughput, validity of inference, task success, average reward over time, etc.—in short, any measurable aspect of the system's operation. *Perturbation tolerance*, then, is the ability of a system to quickly recover—that is, to re-establish desired/expected performance levels—after a perturbation. To achieve this, a perturbation-tolerant system should not only notice when it isn't behaving how it ought or achieving what it should, but be able to use this knowledge to make targeted alterations to its own modules. Such changes can be as simple as re-calibrating its sensors, or as complex as training new (or retraining old) behaviours, changing its rules of inference, learning new words and concepts, adopting different basic ontologies in different circumstances, and even adapting to new notational conventions and recognizing and fixing typographical errors (e.g. misspellings, missing parentheses).[1]

[1]These latter examples present a difficulty for *any* KR system; for whatever KR is used, incoming data might use a

Vol. 15 No. 1,
doi:10.1093/logcom/exh034

While it may often be possible to anticipate the kinds of problems a system will face over its lifetime, and build in specific mechanisms to handle these issues, we doubt this will prove, in the long run, to be the most effective strategy. We believe, in contrast, that efforts should be aimed at implementing mechanisms that help systems help themselves. The goal should be to *increase* their agency and freedom of action in responding to problems, instead of limiting it and hoping that circumstances do not stray from the anticipations of the system designer. We should be creating self-aware, self-guided learners. Indeed, we believe that such metacognitive skills are the key to achieving near-human-level (or, indeed, any useful kind of) non-brittleness. Metacognitive learners would be advanced active learners, able to decide what, when, and how to learn. This will allow systems the needed autonomy to function in domains where human supervision cannot be constantly supplied.

Our general strategy in working toward this goal has been to equip artificial agents with the ability to *notice* when something is amiss, *assess* the anomaly[2], and *guide* a solution into place. We call this basic strategy of self-guided learning the *metacognitive loop* (MCL); it involves the system monitoring, reasoning about, and, when necessary, altering its own decision-making components. This is, after all, what people do, and do well.[3] Indeed, in our view this is largely what perturbation-tolerant commonsense reasoning consists in, rather than in finding special clever solutions to thorny problems. An MCL-based system knows what it is attempting to do, so that it can determine when things are not going well, instead of blindly following its programming over the proverbial cliff—as did one of the DARPA Grand Challenge entries which kept trying to drive through a fence it could not see. If that system had known it was supposed to make forward progress and noticed that it was not doing so, this would have been the first step to overcoming the problem. Or consider the case of a satellite given the command to turn and look at some object away from Earth, but not told to turn back to Earth when finished. Once the satellite turned, there was no way to feed it further commands, and the satellite was lost. In contrast, a system that had general expectations for its operation (frequent communication from Earth), based on the sort of system it was, might have been able to use this knowledge to recover from such mistakes.

For performance in the face of unexpected perturbations can be enhanced even when one cannot figure out exactly what is wrong, or what to do about it, so long as one is able to realize that *something* is wrong, and ask for help, or use trial-and-error, or even give up and work on something else. In our ongoing work, we have found that including an MCL component can enhance the performance of—and speed learning in—different types of systems, including reinforcement learners, natural language human–computer interfaces, commonsense reasoners, deadline-coupled planning systems, robot navigation, and, more generally, repairing arbitrary direct contradictions in a knowledge base.

---

different system, or have typos. How then can a fixed KR system meet the challenge of usefully representing data not 'properly' expressed in that system? The work presented here is, in part, an attempt to remedy this. To foreshadow what is to come: we will describe a *flexible* KR intended to be able to reshape its own notation and its own interpretations of the notation, among other capabilities.

[2] We define an anomaly as a deviation from expected values or outcomes.

[3] In fact, there is some empirical evidence for the importance of metacognition in dealing with the unexpected or unfamiliar. In studies of human learning strategies, it has been established students preparing for—or taking—an exam will make judgments about the relative difficulty of the material to be covered, and use this to choose study strategies, or which questions to answer first. Not surprisingly, in these cases, accuracy of metacognitive judgments correlates with academic performance [43, 42]. Moreover, neurophysiological findings indicate that the frontal lobe has specialized responsibility for metacognitive behaviour [41]. For instance, patients with frontal lobe damage have trouble handling a 'reversal shift', which involves (a) recognizing that a word or concept one has learned to apply to, say, big things, is now being used by others to refer to small things, and (b) making the appropriate adjustment [30, 31].

2 Three problems in commonsense reasoning

Over a period of years, in working towards the design and implementation of (a high degree of) artificial commonsense reasoning, we have struggled to overcome three obstacles in particular, which we refer to by the nicknames of slippage, KR mismatch, and contradiction. *Slippage* is, simply, the divergence between what is believed at a given time, and what has changed at a later time. Of course, the basic problem of belief revision—of rationally revising one's beliefs in light of new evidence—is a widely studied issue [26, 27]. However, in our view the problem bites deeper than is generally supposed, for in all cases, at a later time, time itself has changed, and this can (and often does) matter a great deal.[4] Consider person A having an appointment to meet person B for lunch at noon, and it is now 11:00am. It is easy enough to represent this as:

EXAMPLE 2.1
$Now(11:00), Lunch(12:00)$

However, whereas the latter formula appears to represent a more or less stable fact, the former is true only for an instant—or perhaps for a minute or so, depending on the time-granularity employed—and that is the point: whatever time it is *now* will not be the time later, and it is that very passage that brings lunch (or any future event) closer. Without this basic fact about the temporal character of the world, planning and acting would be meaningless. Thus we need a way to allow $Now(11:00)$ to be updated again and again, and to have that evolution of Now to play a central role in reasoning, so that, for instance, when $Now(11:45)$ holds (or is believed) the agent will begin walking to the agreed-upon location for lunch.

The *KR mismatch* problem is this: there are indefinitely many ways to represent a given circumstance, and systems using one set of representational conventions may not be able to recognize expressions cast using a different set. We consider this a special instance of the more general problem of language meaning, for an expression typically is presented in order to convey a meaning, that is, to cause a certain belief state in the presentee, including beliefs about what the presentee should do; and it is typically this meaning, rather than the expression itself, that needs to be used in the ongoing reasoning process (although, of course, sometimes it is necessary to reason *about* the expression in order to realize or appreciate its meaning). But this requires being able to separately represent and reason about the *meaning* of an expression, and its *form*, that is, it requires taking a meta-linguistic perspective, and the ability to treat words as objects. This is related to the use–mention distinction [10], as well as to work on recognizing utterer intentions; but for us the central question is: how can one recognize the (or a useful) meaning for an expression when the expression is not already known (e.g. when it is not already part of the KR system in use)? This includes cases of new expressions that need to be added to the system, as well as errors (e.g. typos) that need to be recognized as deviations from the existing KR.[5]

The *contradiction* problem is this: how to reason effectively in the presence of contradictions? Much work has been done related to this, especially in the guise of *paraconsistent logics* [49, 48]. Indeed, it is customary to define a paraconsistent logic as one in which the presence of a contradiction need not entail all sentences in the language. The standard paraconsistent approaches address

[4] In addition, standard approaches to belief revision, based on classical logic, do not address the capacity of reasoning with contradictions. But any real agent will have inconsistent beliefs, and so needs the ability to reason in the presence of inconsistencies. See below. Note that our approach to this issue does not require the assumption that new information is more reliable than, or will necessarily replace, beliefs currently in the KB.

[5] The mismatch problem is related to 'fast mapping', the ability to learn words from a single instance of use [13]. In earlier writings we have used 'rapid semantic shift' to include both fast mapping and real-time disambiguation and/or correction of meanings.

contradictions by side-stepping any inconsistencies and reasoning only with consistent portions of the KB. Our view is a bit different: contradictions in one's KB are inevitable [45, 46], and there is no safe haven from which to address them. One must reason with the contradictions as best one can, replacing and repairing beliefs, like the planks of Neurath's boat, one by one while *en route*. Indeed, our work strongly suggests that contradictions are generally useful (so long as they are discovered), in that they point to issues and problems that need to be addressed.

We have found that all three problems lend themselves to a uniform treatment, namely the metacognitive loop (in which time plays a central role). This brings us to an underlying formal basis for our work, namely active logic.

3 Active logic: time-situated commonsense reasoning

Our formal approach to effective reasoning in the presence of slippage, KR mismatches, and contradictions—active logic—is motivated in part by the observation that all reasoning takes place step-wise, in time.[6] This allows an agent to maintain control over, and track, its own reasoning processes. As will be seen, active logic is a type of paraconsistent logic, albeit rather different from standard ones. An account of the basic concepts can be found in [22]. We are also working on a formal semantics, the first results of which are to be reported in [5].

In active logic, aspects of the environment are represented as first order formulas in the knowledge base. Such formulas might represent perceptions of a user's utterance, observations about the state of the domain, or rules added by a system administrator. Inference rules provide the mechanism for 'using' the knowledge for reasoning.

Each 'step' in an active logic proof itself takes one active logic time-step; thus inference always moves into the future at least one step and this fact can be recorded in the logic. In fact, to achieve much of their reasoning, active logics employ a notion of 'now' that is constantly updated by the 'clock rule' shown in Example 3.1.

EXAMPLE 3.1

$$\begin{array}{rl} \text{i:} & Now(i) \\ \hline \text{i+1:} & Now(i+1) \end{array}$$

The clock rule states that from the fact that it is step i at the current step, the step number of the next step is $i + 1$. This step-wise tracking model of time is very different from the 'time-frozen' characterization of time that temporal logic [2, 52] has. The notion of past, present and future, that temporal logics have do not change while theorems are being derived. This sharply contrasts with the special evolving-during-inference model of time that active logics have. When an agent is reasoning about its own ongoing activity, or about another agent whose activity is highly interdependent, traditional 'time-frozen' reasoning is at a disadvantage, and 'time-tracking' active logics can bring new power and flexibility to bear. For instance, theorems can be marked with their time (step-number) of being proven, i.e. the current value of 'now'. This step-number is itself something that further inferences can depend on, such as inferring that a given deadline is now too close to meet by means of a particular plan under refinement if its enactment is estimated to take longer than the (ever shrinking) time remaining before the deadline.

What this means more generally is that, for active logic, beliefs are held at times, and the KB is therefore considered to be a temporally embedded and evolving set of formulas. Thus, the meaning

[6] Other approaches to commonsense reasoning incorporating this basic insight include [12, 32]. Labelled deductive systems (see [23]) appear to provide a generalization of at least some aspects of active logic.

of an inference rule such as that shown in Example 3.2 (an active logic analogue to *modus ponens*), is that if A and $A \rightarrow B$ are in KB at time (step number) i, then B will be added to the KB at time $i + 1$.

EXAMPLE 3.2

$$\begin{array}{rl} \text{i:} & \text{A, A} \rightarrow \text{B} \\ \hline \text{i+1:} & \text{A, A} \rightarrow \text{B, B} \end{array}$$

Although in active logic the logical consequences of the evolving KB do not become *part* of the KB until they are actually derived, inheritance rules ensure that, once derived (or otherwise added to the KB), formulas are carried forward and persist over time.[7] By default, all beliefs from one step that are not directly contradicting are inherited to the next step. This allows the representation of persisting facts or states-of-affairs. However, some beliefs, like the ones related to the current time, are not inherited to the next step; note, for instance, that in the clock rule (Example 3.1) the belief $Now(i)$ is not inherited at step $i + 1$. One simple version of such an inheritance rule, which also illustrates the use of firing conditions, is shown in Example 3.3:

EXAMPLE 3.3

$$\begin{array}{rl} \text{i:} & \text{A} \\ \hline \text{i+1:} & \text{A} \end{array}$$

condition: $\neg$A $\notin$ KB at step i and A $\neq$ Now(i)

Thus, to bring all this together, let us re-consider our lunch example. As noted above in Example 2.1, person A knows the current time, knows to meet person B at noon, and knows to leave for the restaurant at 11:45. We can represent this knowledge, and the deductive process required to get person A to leave on time, in terms of the following active logic inference (with the new beliefs at each step indicated in bold):

EXAMPLE 3.4

$$\begin{array}{rl} \text{11:15 :} & \text{Now(11:15), Meet(B,Lunch,12:00),} \\ & \text{Now(11:45am)} \rightarrow \text{Go(Lunch)} \\ \hline \text{11:16 :} & \textbf{Now(11:16)}\text{, Meet(B,Lunch,12:00),} \\ & \text{Now(11:45am)} \rightarrow \text{Go(Lunch)} \\ \hline \ldots\text{:} & \ldots \\ \hline \text{11:45 :} & \textbf{Now(11:45)}\text{, Meet(B,Lunch,12:00),} \\ & \text{Now(11:45am)} \rightarrow \text{Go(Lunch)} \\ \hline \text{11:46 :} & \textbf{Now(11:46)}\text{, Meet(B,Lunch,12:00),} \\ & \text{Now(11:45am)} \rightarrow \text{Go(Lunch), } \textbf{Go(Lunch)} \end{array}$$

Note that all the beliefs except the time are inherited, and that the rule $Now(11:45) \rightarrow Go(Lunch)$, although it fires at 11:45, does not produce its conclusion until the next time step.[8]

In addition to the formulas obtained from applying rules of inference to formulas at the previous step, new formulas can be added at each step. Step-wise reasoning, coupled with this ability to add

[7] Inheritance and disinheritance are directly related to belief revision [25] and to the frame problem [36, 18]; see [44] for further discussion.

[8] Of course, production active logic systems are much faster than one deduction per minute!

new formulas, ensures that the logic would not get stuck in a lengthy proof, oblivious of the other events that occur *during* the reasoning.

It is the time-sensitivity of active logic inference rules that provides the chief advantage over more traditional logics. Thus, an inference rule can refer to the results of all inferences *up until now*—i.e. thru step i—as it computes the subsequent results (for step $i+1$). This allows an active logic to reason, for example, about its own (past) reasoning; and in particular about what it has *not* yet concluded. Moreover, this can be performed quickly, since it involves little more than a lookup of the current knowledge base.

As mentioned already above, since in active logic the notion of inference is time-dependent, it follows that at any given time only those inferences that have actually been carried out so far can affect the present state of the agent's knowledge. As a result, even if directly contradictory wffs, P and ¬P, are in the agent's KB at time i, it need not be the case that those wffs have been used by time i to derive any other wff, Q. Indeed, it may be that i is the first moment at which both P and ¬P have simultaneously been in KB.

By endowing an active logic with a 'conflict-recognition' inference rule such as that in Example 3.5, *direct* contradictions can be recognized as soon as they occur, and further reasoning can be initiated to repair the contradiction, or at least to adopt a strategy with respect to it, such as simply avoiding the use of either of the contradictands for the time being. Unlike in truth maintenance systems [20, 21] where a separate process resolves contradictions using justification information, in an active logic the contradiction detection and handling occur in the same reasoning process [38]. The `Contra` predicate is a meta-predicate: it is about the course of reasoning itself (and yet is also part of that same evolving history).

EXAMPLE 3.5

$$\begin{array}{rc} \text{i:} & \text{P, } \neg\text{P} \\ \hline \text{i+1:} & \text{Contra(i, P, } \neg\text{P)} \end{array}$$

The idea then is that, although an indirect contradiction may lurk undetected in the knowledge base, it may be sufficient for many purposes to deal only with direct contradictions. Sooner or later, if an indirect contradiction causes trouble, it may reveal itself in the form of a direct contradiction. After all, a real agent has no choice but to reason only with whatever it has been able to come up with *so far*, rather than with implicit but not yet performed inferences. Moreover, since consistency (i.e. the lack of direct or indirect contradictions) is, in general, undecidable, all agents with sufficiently expressive languages will be forced to make do with a hit-or-miss approach to contradiction detection. The best that can be hoped for, then, seems to be an ability to reason effectively in the presence of contradictions, taking action with respect to them only when they become revealed in the course of inference (which itself might be directed toward finding contradictions, to be sure).

Thus, the trick to detecting and dealing with contradictions is to look backward at what one's reasoning has been, rather than forward to what it might be (as traditional automated commonsense or nonmonotonic reasoning formalisms do [1, 24]). Thus at time-step $i+1$ our systems look at what was in their KB at step i and earlier, e.g. to infer that there was a (direct) contradiction at step i [39]; or that it is now too late to meet a deadline given what has been accomplished so far (by step i) and given what remains to be done [44]; or that a particular word is not recognized [7]. Such looking backward appears to provide a computationally feasible handle on perturbation tolerance, allowing an automated reasoner to note and assess anomalies and alter its ongoing courses of reasoning and action accordingly, much as a human appears to do.

Interestingly, having in place these mechanisms for managing contradictions makes dealing with the problems of KR mismatch and slippage much easier. Thus, for instance, adding and/or chang-

ing formulas in the KB need pose no special problems, nor require any expensive (and ultimately undecidable) consistency checks; this makes addressing the slippage problem, by having one's KB change over time, relatively straightforward. Likewise, once one has accepted the notion that one is limited to dealing only with currently derived formulas in the KB—and not also with all the eventual consequences of those beliefs—negative introspection, i.e. knowing what is not known, amounts to a simple search in the KB for a given formula. More generally, the ability to make assertions about the contents of the KB (such as what it does not contain), or about particular beliefs (e.g. that they are suspect) is the first necessary step to being able to reason not just with, but *about* one's own knowledge. This is crucial to dealing with the mismatch problem, for when dealing with language and meaning it is often necessary to recognize and represent the difference between the form and the meaning of an expression [10], for instance to assert that two words mean the same thing, or that one doesn't know the meaning of a given expression.

These temporal and metacognitive aspects make active logic systems more flexible than traditional AI systems and therefore more suitable for reasoning in noisy, dynamic and inconsistent environments, and thus it has proved a very good basis for the development of MCL.

4 From active logic to MCL

Some years ago it was a popular notion that there were two major competing AI methodologies: the 'neat' and the 'scruffy', representing—roughly and respectively—symbol-laden software (with a relatively clear semantics) and adaptive software (that could be tweaked until it 'worked'). For instance, one important 'scruffy' approach, on which we hope to improve, is Brooks' behaviour-based robotics [14, 16]. Brooks suggests that sophisticated robotic intelligence can and should be built through the incremental addition of individual layers of situation-specific control systems. The only direct interaction between layers is through the suppression or activation of certain pathways (in rough analogy to the workings of neural systems). However, all layers have access to inputs from perception (although they respond only to those elements to which they are specifically attuned) and can control certain aspects of the robot's behaviour, which offers a great deal of indirect, environmentally mediated interaction between the layers. Although we agree that an architecture of this sort can provide fast and fluid reactions in real-world situations, we cannot accept Brooks' claim [17, 15] that such an approach can ever achieve the flexibility and robustness of human intelligence (for some arguments to this effect, see [33, 4]). For that, in addition to 'scruffy' systems providing fast and fluid *reactions*, there must be 'neat' systems supporting both *deliberation* and *re-consideration*, which we have argued calls for symbolic reasoning and (most importantly) *meta*-reasoning, capable of self-monitoring and self-correction [6, 11, 19, 47]. That is to say, we think that a fast, fluid and *flexible*—i.e. non-brittle—real-world system can be achieved by adding a layer of symbolic (meta-) reasoning on top of adaptive control layers, and allowing it not just to *suppress* the adaptive layers, but also to *re-train* them when necessary.

Of course, the basic idea of combining 'neat' and 'scruffy' approaches is not new, but while it has long been recognized that both methodologies are important and need to be combined, efforts along those lines to date (e.g. Ron Sun [56, 61, 55, 54], Ofer Melnik and Jordan Pollock [37], and Gary Marcus [34]) have—in our view—overlooked the most exciting advantage to be gained from a proper joining. Our contention is that a *triadic* architecture will be able to cut through the brittleness barrier, which, as we asserted above, is perhaps the single most pervasive problem in AI research. Our suggested triad is comprised of (1) trainer module(s), (2) trainable modules (many of which may perform symbolic/reasoning computations), and (3) an oversight module that executes the metacognitive loop (MCL). Note that symbolic modules may be in as much need of re-

tuning as may traditionally adaptive modules; and conversely, symbolic processing may be critical in the effective adaptation of the latter. Thus we postulate the special (symbolic) MCL module that oversees both of these. Consequently, in our approach there is less distinction between symbolic and adaptive modules; (almost) everything may adapt via MCL (executed by an exceptional non-adaptive module).[9]

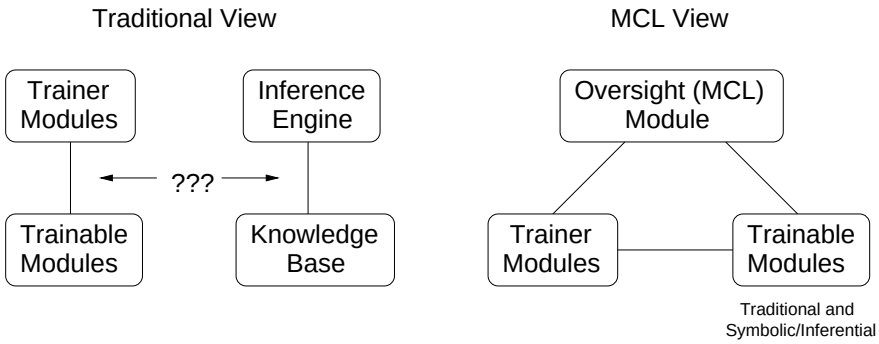

FIGURE 1. Traditional and triadic-MCL views of the relationship between symbolic and adaptive processing

In traditional work, adaptive and symbolic processing were not brought together, or if so then usually only peripherally; in rare exceptions the symbolic aspects were put into the adaptive portion, but only to show out that symbols, too, can be implemented in (say) distributed ways, as they presumably are in the brain. But in the proposed view MCL decides when a trainer should initiate (or stop) adaptation in another trainable (whether symbolic or not) module, and MCL may also, if the matter is simple enough, carry out the adaptation directly. MCL may even decide that a new trainable module is to be created, if existing ones do not seem close to being able to address the issue at hand.

As mentioned already above, active logic has the necessary features—most importantly time sensitivity and contradiction-tolerance—to implement MCL in real-world systems. The overall idea is as follows: errors (contradictions, typos, mismatches between world and word, missed deadlines, etc.) and other kinds of perturbations occur. How does one detect and reason about such perturbations? As we have defined the terms, a perturbation will cause an anomaly, that is, a deviation from the expected performance of the system. Thus, we have found in our work to date, e.g. see [50, 46, 58, 7] that a very wide range of anomalies are readily expressed in terms of contradictions. This is quite clear in some cases, e.g. when two normally trusted sources report conflicting data. But it also seems to work well in others, such as Expected(A) and Observed($\neg$A), as long as the expectation has led A to be asserted into the KB, so that the observation of $\neg$A contradicts it.[10] Thus MCL crucially involves the generation of expectations for performance, as well as for the outcomes

[9]The possibility of letting the MCL module *itself* be trained for improvement is intriguing; however it is beyond the scope of this paper.

[10]This is akin to McCarthy's use of abnormalities [35]. However, while that approach can lead to thorny problems requiring prioritized circumscription in more traditional settings, for us the real-time character of active logic allows the system the freedom to eventually decide to ignore such complex cases if they are not resolved easily. Thus a key notion for us is that such rather shallow inferences are sufficient for MCL to greatly improve overall performance.

of specific actions, and the continual monitoring of the KB for contradictory pairs.

For a simple illustration, consider our lunch example from the perspective of person B, already waiting at the restaurant. Person B expects to meet person A at 12:00; but let us suppose that person A is late. We might get a series of inferences like the following:[11]

EXAMPLE 4.1

11:59 :	Now(11:59), ¬See(A), Expect(See(A), 12:00), (Expect(x,t) & Now(t)) → Assert(x)
12:00 :	**Now(12:00)**, ¬See(A), Expect(See(A), 12:00), (Expect(x,t) & Now(t)) → Assert(x)
12:01 :	**Now(12:01)**, ¬See(A), Expect(See(A), 12:00), (Expect(x,t) & Now(t)) → Assert(x), **Assert(See(A))**
12:02 :	**Now(12:02)**, ¬See(A), Expect(See(A), 12:00), (Expect(x,t) & Now(t)) → Assert(x), **See(A)**
12:03 :	**Now(12:03)**, Expect(See(A), 12:00), (Expect(x,t) & Now(t)) → Assert(x), **Contra(12:02, See(A),¬See(A))**

When person A doesn't show up on time, this generates an anomaly, in the form of a contradiction. Once such an anomaly is detected, it is compared to a stored (but dynamically changing, as learning proceeds) list of anomaly-types. A match then provides access to a second list (again changeable) of options for dealing with (repairing) that type. A choice is made among these (e.g. based on time and other factors). If no match is found, a fall-back option is used (ask for help, trial-and-error, put on hold, give up, ignore). In the current case, person B might note that the anomaly is a missed appointment, and pick the first option there, namely Phone(A).

After an option is chosen, MCL attempts to guide it into place ('make it so'). In some cases this is easy and MCL can carry out the entire repair, e.g. inserting a new belief into the KB, if that is the repair. Other cases may not be so easy, e.g. it may be necessary to call an external process (as in the above case of making a phone call), or even to retrain a module, such as memory retrieval, or a neural net that controls locomotion, or ask for advice about a possible new category and await the response (for further examples of anomalies, and discussion of response options, see Sections 5 and 6, below).

MCL must be kept simple and fast; it is not aimed at clever tricks or deep reasoning. This is important, so that the system (and MCL in particular) does not get bogged down in its own efforts. As pointed out earlier, if it takes too long on something, it must notice that and make a decision as to whether to give up on it, or try another tack. Active logic was designed with this general kind of time-sensitive capability in mind, and has been successfully applied to similar situations before (e.g. deadline-coupled planning, mentioned above).

Moreover—and crucially—active logic provides a mechanism to note and forestall 'bogging-down'. That is, suppose MCL happens to encounter more and more anomalies (perhaps generated as recursive calls to its guiding step (iii), if that guidance is getting nowhere). Then the passage of sufficient time with no noted progress on a given goal will itself trigger an anomaly that will force a resolution of work on that goal, either to abandon it, postpone it, or seek help. The 'sufficient' amount of time can even be reset by the logic, given data on the relative importance of progress

[11] In this example, ¬ See(A) would be put into the KB from the perception system. While one does not necessarily want the default operation of the perceptual system to be to continually assert everything it does not see, it *can* be useful to specifically assert that one doesn't see something one is actively looking for.

toward goals. This is one important advantage that time-tracking meta-reasoning has over other approaches to meta-reasoning, such as [53], that assume that meta-reasoning will take little time, and thus have no built-in mechanism for handling situations when that assumption is violated.

Several studies of ours illustrate the benefits of meta-reasoning and metacognitive monitoring in improving overall system performance, and decreasing brittleness, both in the case where the system consists entirely of symbolic reasoners and in the case where a symbolic meta-reasoner monitors and corrects a neural net or a reinforcement learner.[12] In the next section, we will outline the results from a few of those projects, after which we will discuss where future work might be directed.

5 Implemented MCL systems

MCL can enhance performance for two related reasons. First, it can monitor and influence on-line performance even without making any basic changes or improvements to action-producing or decision-making components. An example of this would be noticing that progress on a task has stopped (i.e. that the system is 'stuck') and directing specific efforts to getting 'un-stuck', or simply moving on to a different task. Second, and more powerfully, MCL can direct the system to actively learn something that it (apparently) doesn't know, or has got wrong. Since there is evidently a great deal that can be learned, depending on the system and the scenario, MCL in this guise is best understood as a principled method of *organizing* and *controlling* learning—deciding whether to learn, what to learn, with what methods, and (importantly) when to stop. An example of this latter ability would be noticing that a problem in processing a given user command appears to be caused by ignorance of a certain word, and taking steps to learn the unknown word.

Both of these abilities are crucial to improving the perturbation tolerance of a given system, and they generally work in concert. Thus, for instance, we have shown that an MCL-enhanced reinforcement learner can—by choosing when to ignore anomalies, when to make minor on-line adjustments, and when to order re-learning of its action policy—always perform *at least as well as*, and in many cases *significantly out-perform*, a standard reinforcement learner when operating in a changing world.

5.1 MCL-enhanced reinforcement learning

In a simple demonstration of this idea, we built a standard reinforcement learner (we tested Q-learning [59, 60], SARSA [57] and prioritized sweeping [40]), and placed it in an 8x8 world with two rewards—reward 1 (r1) in square (1,1) and reward 2 (r2) in square (8,8). The learner was allowed to take 10,000 actions in this initial world, which was enough in all cases to establish a very good albeit non-optimal policy. In turn 10,001, the values of the rewards were abruptly changed. See [9] for a complete account of the experimental design and results.

We found that the perturbation tolerance (i.e. the post-perturbation performance) of standard reinforcement learners was negatively correlated to the degree of the perturbation—the bigger the change, the worse they did. However, even a simple (and somewhat stupid) MCL-enhancement, that did no more than generate and monitor expectations for performance (average reward per turn, average time between rewards, and amount of reward in each state) and re-learn its entire policy whenever its expectations were violated three times, outperformed standard reinforcement learning in the case of high-degree perturbations. And, as already mentioned, a somewhat smarter MCL-enhancement, that chose between the available methods of doing nothing, making an on-line adjustment, and re-

[12] See, for instance, [50, 51, 6, 19, 11, 46, 28, 9]; most of these have used active logic to provide the reasoning and meta-reasoning component of the overall system.

learning its policy, in light of its assessment of the anomalies, performed best overall, (see figure 2, 'sophisticated-MCL'), despite some under-performance of this version of MCL in response to mid-range perturbations.

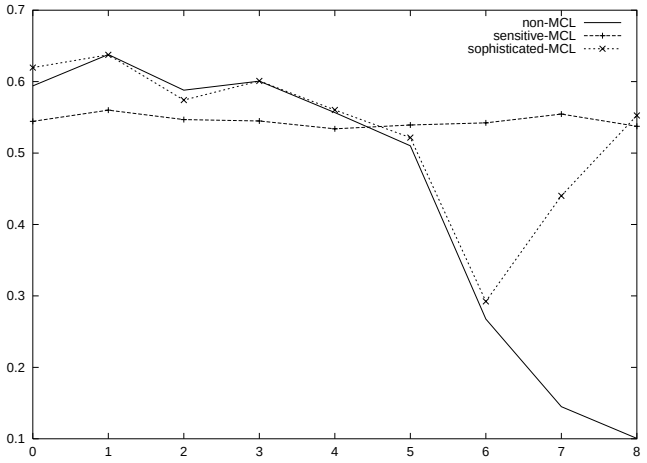

FIGURE 2. Post-perturbation performance of standard Q-learning, sensitive-MCL and sophisticated-MCL, as a function of degree of perturbation

In a manner not too different from that illustrated in Example 4.1, sophisticated-MCL generates and records expectations, based on ongoing experience, for the average reward it will get per turn, average time between rewards, and the amount of reward expected in each system state. When its experience deviates from these expectations, it uses the anomalies to determine whether, how, and to what degree the world has changed. Thus, for instance, in the case of an anomaly such as an increased time to reward, its initial assumption is that a local problem in the action policy was causing the system to 'cycle' between states (i.e. that it was stuck), and its response is to temporarily increase the exploration factor so as to break out of the cycle. However, it continues to monitor the situation, and if this does not solve the problem, or if this anomaly occurs along with significant violations of its expectations for the values of rewards, then the system decides that the world had changed significantly, and, rather than continue to make small on-line adjustments, it orders its action policy to be re-learned.

5.2 MCL-enhanced navigation

Another agent that we have been developing uses a neural net for navigation; however it also has a monitoring component that notices when navigational failures (such as collisions) take place, and records these and their circumstances. It is then able to use this information to assess the failures and make targeted changes to the neural net, including starting with a different set of weights, or re-training on a specific set of inputs. The agent exhibits better behaviour while training, and also learns more quickly to navigate effectively [28].

Although both the above systems are relatively simple, they do illustrate the ways in which self-monitoring and control can help systems maintain performance in the face of changes, and, more particularly, they demonstrate cooperation between the ability to initiate new actions, and the ability

to initiate new learning. Still, one has to expect that as the scenarios, and the systems themselves, become more complex, more sophisticated and expressive reasoning mechanisms will be required to usefully assess and appropriately respond to anomalies. Thus we turn in our final example to the very representation-rich and difficult domain of natural-language human-computer interaction.

5.3 MCL-enhanced human-computer dialogue

One of the most important application areas for active logic has been natural language human–computer interaction (HCI). Natural language is complex and ambiguous, and communication for this reason always contains an element of uncertainty. To manage this uncertainty, human dialogue partners continually monitor the conversation, their own comprehension, and the apparent comprehension of their interlocutor. Both partners elicit and provide feedback as the conversation continues, and make conversational adjustments as necessary. The feedback might be as simple as 'Got it?', eliciting a simple 'yes', or as complex as 'Wait. I don't think I understand the concept of hidden variables', which could result in a long digression. We contend that the ability to engage in such meta-language, and to use the results of meta-dialogic interactions to help understand otherwise problematic utterances, is the source of much of the flexibility displayed by human conversation [47]. Although there are other ways of managing uncertainty (and other types of uncertainty to be managed), we have demonstrated improved performance can be achieved by enhancing existing HCI systems with the ability to engage in meta-reasoning and meta-dialogue.

One earlier achievement was the design and implementation of a model of action-directive exchanges (task oriented requests) based on the active logic model of inference. Our model works via a step-wise transformation of the literal request made by a user (e.g. 'Send the Boston train to New York') into a specific request for an action that can be performed by the system or domain. In the case of 'the Boston train', the system we have implemented is able to interpret this as 'the train in Boston', and then further disambiguate this into a specific train currently at Boston station, which it will send to New York. Information about each step in the transformation is maintained, to accommodate any repairs that might be required in the case of negative feedback (if for instance, the system picks the wrong train, and the user says 'No' in response to the action). This implementation represents an advance not just in its ability to reason initially about the user's intention (e.g. by 'the Boston train' the user means . . .) but in its ability to respond in a context-sensitive way to post-action user feedback, and use that feedback to aid in the interpretation of the user's original and future intentions. For instance, in one specific case tested, the user says 'Send the Boston train to New York' and then, after the system chooses and moves a train, says 'No, send the *Boston* train to New York'. Such an exchange might occur if there is more than one train at Boston station, and the system chose a train other than the one the user meant. Whereas the original TRAINS-96 dialogue system [3] would respond to this apparently contradictory sequence of commands by sending the very same train, our enhanced HCI system notes the contradiction, and, by assessing the problem, identifies a possible mistake in its choice of referent for 'the Boston train'. Thus, the enhanced system will choose a different train the second time around, or if there are no other trains in Boston, it will ask the user to specify the train by name. The details of the implementation, as well as a specific account of the reasoning required for each of these steps, can be found in [58].

A more recent advance we have made along these lines, in a system we call ALFRED [13], was to enhance the ability of our HCI system to more accurately assess the nature of dialogue problems, and to engage in meta-dialogue with the user to help resolve the problem. For instance, if the user says 'Send the Metro to Boston', the original system would have responded with the unhelpful fact

[13] **A**ctive **L**ogic **F**or **R**eason **E**nhanced **D**ialogue.

that it was unable to process this request. Our system, in contrast, notices that it doesn't know the word 'Metro', and will instead request specific help from the user, saying: 'I don't know the word "Metro" '. 'What does "Metro" mean?' Once the user tells the system that 'Metro' is another word for 'Metroliner', it is able to correctly implement the user's request [8, 7, 29]. It can use these same methods to learn new commands, so long as the new command can be explained in terms of (including being compounded from) commands it already knows.

Two key features that support these behaviours are a rule for detecting contradictions (which is a standard feature of active logic), and the ability to set and monitor expectations. If the agent notices that an expectation has not been achieved, this causes the agent to assess the problem and consider the different options it has to fix it. This includes the detection of time-related anomalies, which are especially important to HCI. For instance, if the user does not respond to a system query with in the expected time limit, then the system recognizes that there is a problem. In this case, the different options that it has to deal with the problem are (a) to repeat the query, (b) to find out from the user whether everything is OK, and (c) to stop expecting a response from the user. The current system initially tries repeating the query. However, continuous repetition of the query without a response from the user indicates a continuing problem (for recall that part of MCL is to monitor the progress of solutions), and causes a re-evaluation of the possible response options. In this case the system would ask the user whether everything is OK. If there is still no response from the user, the system will drop its expectation about getting a response from the user in the near future.

Equally important to the above abilities, however, is an enhancement to ALFRED's representational scheme allowing it to differentially represent the various aspects of words and language, e.g. extension (reference), intension (concept), orthographic form (spelling), and the like. This gives ALFRED the flexibility and expressive power to come to meta-level conclusions, such as that two different words (different orthographic forms) have the same referent, or vice versa, or that it doesn't know anything about a given word *but* its orthographic form, and so must learn more about it.

6 Future work

Each of the systems described above relies for its robust behaviour on the relatively simple expedient of generating, and monitoring for violations of, expectations for overall performance, and for the outcome of each action taken. As we have already shown, immediate performance improvements can be expected by virtue of this step alone, if only by allowing systems to notice when something is going very wrong and stop (or do something else). Indeed, it has been suggested to us that the MCL approach—and, more generally, the design of systems able to generate and monitor expectations for their own performance—will allow more problems to be caught at the design stage, as careful thought must be given to what, exactly, the system should expect in each system state. However, clearly the most powerful and promising aspect of the research lies in the further steps of allowing the system to assess the anomalies and guide into place a targeted response, one that can include self-directed active learning.

This suggests three major research questions, roughly corresponding to the three stages, *note-assess-guide*, of the MCL paradigm.

1. What kinds of expectations, at what levels of the system architecture, are most important to track, and how should these be represented so as to make anomalies readily noted? More generally: how much, and what kind of self-knowledge, represented in what form, is required to support the abilities envisioned? In our work on reinforcement learning, for instance, we found that average reward per turn was not a particularly useful metric, but that time between rewards was an extremely important indicator of on-line performance.

2. What methods might be used to accurately assess anomalies, as part of the effort to decide upon an appropriate response? In partial answer to this question, we have been developing a general typology of contradictions [46], but there are also likely to be important situation- and architecture-specific indicators that need to be identified. Thus, for instance, in the reinforcement learning work, it was discovered that valence changes in the rewards was a good indicator of the need to re-learn the action policy.
3. What strategies are most effective for guiding responses into place? Here again, there are likely to be some generic and/or fall-back strategies useful in very many situations (e.g. get more information, ask for help, move on to a different task), and also some situation- and architecture-specific responses, including various kinds of learning. With respect to reinforcement learning, we are investigating the efficacy of such methods as self-shaping (having the system define a shaping function in light of its partial knowledge of its new environment) and directed exploration. More generally, we are exploring such methods to improve learning as mistake-driven boosting and dynamic alteration of bias.

6.1 A near-term project for MCL

These research questions are, of course, very broad, and could be pursued in any number of ways. To make the discussion more concrete, and also as a way of explaining our particular research agenda, let us imagine an autonomous search-and-rescue vehicle, that could enter an unknown structure, traverse it, find all the people inside, and, where necessary, return to specific individuals with items such as food, water, and the like—a kind of autonomous, urban, St. Bernard (AUSB). Even putting aside some of the difficult physical challenges presented by such a domain (climbing over rubble, or up and down stairs and steep inclines), there is plenty of opportunity for environmental and system perturbation: passageways can become blocked, people can move, apparent survivors can perish, and apparent casualties can turn out to be alive, lighting and visibility conditions can change (over time, and in different parts of the structure), and the AUSB itself can become damaged (e.g. by falling debris, fire or water) and need to adjust accordingly.

Naturally, of primary concern for such a system would be implementing robust and flexible navigation and mapping abilities, and there are many standard approaches to this problem. How would the MCL approach be different from, or improve upon any of these? Well, one generally recognized drawback of the standard approaches to, for instance, obstacle detection and avoidance is that they are generally tuned to detect and react to specific kinds of obstacles—i.e. things that look a specific way to their sensors. Our approach, in contrast, is to enhance the standard methods with MCL, consistently comparing expectations for performance, based on how things look to the system, with actual performance. Divergence of the two—as would occur, for instance, in the case where no progress is being made even though the sonar indicates no obstacles, or when a previously developed map indicates a passageway where there is now an obstacle—would trigger appropriate recovery procedures. The immediate benefit would be a building-navigating robot that would flexibly respond to changes in its environment, and never get permanently stuck—at least to the extent that it would *notice* if it were stuck, or going in unintended circles, and initiate a recovery-response process.

Another necessary feature for such a system would be person-detection capabilities. Here again, the advantage that MCL would provide to such a system is not a new and better technique for detecting people, but rather a method for best taking advantage of *current capabilities*. For consider that it is unlikely that all person-detecting techniques are equally useful in all situations, and also unlikely that one can know with certainty, in advance, which ones will work better when. So, for

instance, in an extremely dusty environment, heat signatures may reveal people even though the vision system is confounded; likewise, in a fire, heat signatures may be less useful than vision, and both less useful than voice-detection. A system which *notices* the fact that it is getting detections on some systems and not others, combined with the fact that some systems don't seem to be operating according to expected norms *in general*, can decide to rely more on some systems than others. The immediate advantage would be an AUSB that flexibly adapts to the situation, dynamically choosing the detection techniques that appear to be most effective in its current circumstances, in light of its *assessment* of anomalies in sensor performance.

Finally, what if the AUSB were equipped with self-guided re-training mechanisms for its learning components, including the pattern-recognition elements of the vision system, and the obstacle-detection and avoidance elements of the navigational system? If the AUSB has to consistently operate under conditions that are not conducive to adequate performance, or if the conditions fluctuate, its systems may have to be re-trained for these new conditions. The system would notice poor performance (and, not incidentally, notice under what conditions it performed poorly, and under what conditions it performed well, which could be important for quick adaptation later on, by quickly switching to a previously learned policy if previously experienced conditions recur) and recalibrate or retrain until performance achieved acceptable levels. How? In addition to the techniques we have implemented already for reinforcement learning and neural-net re-training, we intend to explore the usefulness of boosting and dynamic alteration of bias—and there are, no doubt, many other possibilities to explore. Here the main advantage would be that online autonomous real-time decisions could be made about which kind of learning to *guide* into place, even when human inputs are not always readily available, as in the case of Mars rovers, undersea exploration, and battlefield reconnaissance. Thus not only will behavioural improvements occur but behaviour-improving *strategies* can be developed autonomously.

6.2 Long-term vision

The example of the AUSB shows how MCL can enhance relatively standard systems, making them more robust and flexible. But it could also be argued that something like MCL, and the metacognitive awareness it implements, is *required* for more sophisticated cognitive adjustments. Thus, imagine that the robot MarSciE[14] is roving the surface of Mars. She is designed to move slowly about the Martian surface, taking and testing samples as she goes. The most exciting possible discovery, of course, would be to find signs of life, and for this she has modules for detecting amino acids and other likely organic compounds. However, her more mundane, primary task is to analyse and classify the types of rocks she comes across, to help scientists better understand Mars' geologic history, and the forces that continue to shape the surface of the planet. She worked very well at NASA's desert test site, classifying rock samples of the sort expected to be found on Mars with over 90% accuracy. But since getting to Mars she has had little to report. The samples she is finding don't seem to fit into the categories with which she has been equipped, and nearly everything is being classified as 'unknown'. It is obvious that there is some kind of problem—obvious, that is, to *us*, but not to MarSciE, who is operating exactly as designed. With no human scientist on site to assess the problem and perhaps re-train MarSciE, the Mars geologic mission is likely to fail.

Solving MarSciE's problem is somewhat complicated, and will surely require the help of scientists back on Earth. That is to be expected; after all, even a human scientist, faced with the unexpected, may well ask for advice. But consider how much could be accomplished by MarSciE on her own, if only she were equipped with MCL. First, just *noticing* there is a potential problem with her mineral

[14] MARs SCIentific Explorer, pronounced 'Marcie'.

classification modules is a big step, for this could trigger an automatic system check: she takes rocks of known classification, and analyses them. The results come back 100% correct. Whatever the problem is, it isn't with the operation or accuracy of those routines. This opens the possibility that the problem lies in a mis-match between MarSciE's classification scheme and Mars' rocks. *Assessing* her options at this point, MarSciE knows she can ask for advice, but she can also try to see if there are any obvious patterns in the data she has so far collected. Since Earth is at this point 22 light-minutes away, making communication slow and difficult, she decides to see what she can learn on her own. Analysing the information from the 'unknown' samples she has collected, she finds they fall into four natural categories. *Now* she is ready for advice. She sends all this data back to Earth and waits.

It turns out that MarSciE's time was well spent. The rocks in one of MarSciE's categories do not correspond to anything known by the scientists. They appear to be a kind of granite-like igneous rock of surprising composition, which they name Gr-M1a. If the classification holds up it could be an important find. The three remaining categories she identified corresponded to known types of rock, which were not expected to be in the part of Mars MarSciE is exploring (thus suggesting a rather different geologic history for the area). However, of these three categories, two actually correspond to the *same* kind of schist. Thus, in light of these findings, the advice she receives is three-fold: first, collapse two of the categories of rock she identified into one. Second, she is to retrain on her samples until she classifies them according to these three new categories (which provisionally appear to be genuine). Third, once she gathers many more examples of Gr-M1a, she is to carefully analyse that category to see if it can either be sub-divided, or merged with any of her other currently known types. As it turns out, the new rock type survives scrutiny, and in honour of MarSciE's role in the discovery, the new rock is named Marcyite. MarSciE duly learns this new name, and classifies her samples accordingly.

It seems fairly clear that autonomous robots with MarSciE's capabilities would be extremely useful; it is also clear that they are beyond the current state of the art. However, we would like to suggest that they are not *so* far beyond the state of the art that it would be impossible to discuss concrete plans for building such a system. For much of what MarSciE needs to be able to do can be done using current technology, or perhaps current technology slightly improved: e.g. she needs to be able to learn categories, and use them in sorting objects according to sensory input, and she needs to be able to take and use advice. Both of these capabilities exist in current systems. What we are suggesting is the primary difference between current systems and a system like MarSciE is the way these components are *organized in*, and *controlled by* the system itself. First there is the simple fact that MarSciE is monitoring her own performance, and comparing it with expectations. This, of course, is simple, and not especially novel. Second is the fact that the system can consider the possibility not just that a given system is *malfunctioning*, but that its representational scheme is somehow inappropriate to the current circumstances. This latter ability is the key metacognitive enhancement in MarSciE that drives the mental flexibility that she displays; MarSciE *knows* or somehow *represents* the fact that she is using concepts and categories, that these concepts may or may not be the most appropriate for a given task, and, most importantly, that they may be changed as a way of possibly improving performance. As mentioned above, we have taken some steps in this direction with ALFRED, but much more work needs to be done on the question of what kinds of representations, coupled with what kinds of reasoning and self-monitoring, are required to support MarSciE's abilities.

More ambitious still is a system we call GePurS[15]. GePurS is used in many domains, so his knowledge base gets very large and complex, with a great deal of knowledge that is at any given moment irrelevant to his current domain. Moreover, many domains require different ontologies and

[15]GEneral-PURpose Scout, pronounced 'Jeepers'.

KR schemes, and in some cases the mis-matches are quite significant. All this slows him down, sometimes so much so that he becomes ineffective—and without MCL he cannot notice his own slowdown, let alone fix it. A programmer has to reorganize the KB, with a working memory (STM) and a long-term store (LTM), as well as retrieval mechanisms; and make sure that the information is compliant with the KR scheme appropriate to the intended domain—and then reprogram it later on when another domain change affects which stored beliefs are relevant, and which ontology is most appropriate.

Here again, we think that the ideal situation is one in which the system itself can notice the domain shifts, or the performance problems that accompany them, and make the necessary adjustments to its systems to cope with each new environment. This will require an even greater degree of self-awareness and self-control than MarSciE had, for whereas she needed to know she was using concepts that can be changed, since GePurS may need to periodically change the way he organizes and utilizes his own KB (including memory, as well as how he treats his own sensor data), *he* will need to represent these facts about himself, and have mechanisms whereby they can be dynamically altered. This goes far beyond changing one's individual concepts, for it can entail not just the shifting of particular items to and from different memory stores, but even the dynamic adoption and adaptation of different ontologies and KR schemes as the domain and necessity dictate.

This would represent a fundamental shift in how KR is currently developed and implemented in intelligent systems: instead of requiring a developer (or, more often, a large team of developers) to get the KR right from the beginning, and then imposing the inflexible scheme on the system, we imagine a process much more similar to teaching a child a language, or training a new employee in the operation of a complex system. Although in these cases the expert trainer may have a detailed and largely stable ontology and KR scheme, the trainee has at best an approximation of this ideal. However, through experience, self-monitoring, recognizing and fixing mistakes, and advice from the trainer, the trainee will come successively closer to an identical, or at least an equally workable, KR scheme.

There are two attributes of such a trainee that are worth trying to reproduce in autonomous systems. The first, and most obvious, is the ability for learning and self-improvement. This requires all the elements of MCL: self-monitoring, the recognition of mistakes and other problems, and the ability to make changes to address these problems. However, equally important, and resting on the very same foundations, is the fact that during the training problems don't (entirely) derail the trainee—he or she can recognize that there is a problem, and take special steps for dealing with it, which, although it may slow down the process a great deal, and eventually result in handing over control to a supervisor, nevertheless does not cause a complete breakdown. Furthermore, most trainees are smart enough to realize that the concepts and procedures they are learning at work don't necessarily apply at home (or, if they don't realize this at first, they may come to this conclusion in light of the various problems that universally using a given scheme can cause). Thus they learn to use one set of concepts and KR scheme at work, and another at home; and they know that both can be adapted as the situation dictates. That is, they are not only able to treat a given KR scheme as flexible, but to flexibly switch *between* KR schemes, all the while noticing, and addressing, problems as they arise. This kind of flexibility *in medias res* is perhaps the most important immediate benefit of the inclusion of MCL in a system.

7 Conclusions

We are the first to admit that building MarSciE will be much more difficult than building the AUSB, and that GePurS is an order of magnitude more complex than MarSciE; and yet we want to insist

that they all lie on the same developmental path, which necessarily involves the implementation of self-monitoring, self-representation, and autonomous self-improvement. However, to follow this path will require some important shifts in how we approach the design and implementation of autonomous, intelligent systems, and how we divide responsibility for the operation and maintenance of the system between the developer and the system itself. This, in turn, suggests that the most pressing areas for future research in intelligent systems are those revolving around the questions outlined above: by what methods, and with what amount of detail, should systems represent themselves, and what should they expect for their own operation; how can the system assess its own failures; what methods are available for fixing those failures, and when should each be used; and, finally, what are the architectures that can support these abilities, and what are the advantages of each.

Acknowledgements

This work is supported in part by grants from AFOSR and ONR. We would like to thank Tim Oates for many fruitful discussions of these ideas, and Ken Hennacy, Darsana Josyula and Waiyian Chong for their roles in the design and implementation of the systems described herein.

References

[1] Logical formalizations and commonsense reasoning, 2004. Special issue of Artificial Intelligence, edited by Ernest Davis and Leora Morgenstern.

[2] J. F. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, **4**, 531–580, 1994.

[3] J. F. Allen, B. W. Miller, E. K. Ringger, and T. Sikorski. A robust system for natural spoken dialogue. In *Proceedings of the 1996 Annual Meeting of the Association for Computational Linguistics (ACL-96)*, pp. 62–70, 1996.

[4] M. L. Anderson. Embodied cognition: a field guide. *Artificial Intelligence*, **149**, 91–130, 2003.

[5] M. L. Anderson, W. Gomaa, J. Grant, and D. Perlis. On the reasoning of real-word agents: toward a semantics for active logic, in preparation.

[6] M. L. Anderson, D. Josyula, Y. Okamoto, and D. Perlis. Time-situated agency: active logic and intention formation. In *Workshop on Cognitive Agents, 25^{th} German Conference on Artificial Intelligence*, 2002.

[7] M. L. Anderson, D. Josyula, and D.Perlis. Talking to computers. In *Proceedings of the Workshop on Mixed Initiative Intelligent Systems, IJCAI-03*, 2003.

[8] M. L. Anderson, D. Josyula, D. Perlis, and K. Purang. Active logic for more effective human-computer interaction and other commonsense applications. In *Proceedings of the Workshop Empirically Successful First-Order Reasoning, International Joint Conference on Automated Reasoning*, 2004.

[9] M. L. Anderson, Tim Oates, Waiyian Chong, and D. Perlis. Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance, in preparation.

[10] M. L. Anderson, Y. Okamoto, D. Josyula, and D. Perlis. The use-mention distinction and its importance to HCI. In *Proceedings of the Sixth Workshop on the Semantics and Pragmatics of Dialog*, 2002.

[11] M. Bhatia, P.Chi, W. Chong, D. P. Josyula, M. Anderson, Y. Okamoto, D. Perlis, and K. Purang. Handling uncertainty with active logic. In *Proceedings of the AAAI Fall Symposium on Uncertainty in Computation*, 2001.

[12] W. Bibel. Let's plan it deductively. *Artificial Intelligence*, **103**, 183–208, 1998.

[13] P. Bloom and L. Markson. Capacities underlying word learning. *Trends in Cognitive Sciences*, **2**, 67–73, 1998.

[14] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, **RA-2**, 14–23, 1986.

[15] R. A. Brooks. Intelligence without reason. In *Proceedings of 12th International Joint Conference on Artificial Intelligence*, pp. 569–95, 1991.

[16] R. A. Brooks. From earwigs to humans. practice and future of autonomous agents, *Robotics and Autonomous Systems*, **20**, 291–304, 1997.

[17] R. Brooks. Intelligence without representation. *Artificial Intelligence*, **47**, 139–60, 1991.

[18] F. Brown, (ed.). *The Frame Problem in Artificial Intelligence*. Morgan Kaufmann, 1987.

[19] W. Chong, M. O'Donovan-Anderson, Y. Okamoto, and D. Perlis. Seven days in the life of a robotic agent. In *Proceedings of the GSFC/JPL Workshop on Radical Agent Concepts*, 2002.

[20] J. Doyle. A truth maintenance system. *Artificial Intelligence*, **12**, 231–272, 1979.

[21] J. Doyle. *A Model for Deliberation, Action, and Introspection*. PhD thesis, Massachusetts Institute of Technology, 1980.

[22] J. Elgot-Drapkin and D. Perlis. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence*, **2**, 75–98, 1990.

[23] D. M. Gabbay and J. Woods. *Agenda Relevance: A Study in Formal Pragmatics*. North-Holland, 2003.

[24] M. Ginsberg, (ed.). *Readings in Nonmonotonic Reasoning*. Morgan Kaufmann, 1987.

[25] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge, MA, 1988.

[26] P. Gärdenfors. *Belief Revision*. Cambridge University Press, Cambridge, 1992.

[27] P. Gärdenfors and H. Rott. Belief revision. In *Handbook of Logic in Artificial Intelligence and Logic Programming, volume IV*, D. M. Gabbay, C. J. Hogger, and J. A. Robinson, eds, pp. 35–132. Oxford University Press, 1995.

[28] K. Hennacy, N. Swamy, and D. Perlis. RGL study in a hybrid real-time system. In *Proceedings of the IASTED NCI*, 2003.

[29] D. Josyula, M. L. Anderson, and D. Perlis. Towards domain-independent, task-oriented, conversational adequacy. In *Proceedings of IJCAI-2003 Intelligent Systems Demonstrations*, pp. 1637–8, 2003.

[30] H. H. Kendler and T. S. Kendler. Vertical and horizontal processes in problem solving. *Psychological Review*, **69**, 1–16, 1962.

[31] H. H. Kendler and T. S. Kendler. Reversal-shift behavior: Some basic issues. *Psychological Bulletin*, **72**, 229–32, 1969.

[32] H. Khalil. *Logical Foundations of Default Reasoning*. PhD thesis, University of Leipsig, Leipzig, Germany, 2002.

[33] D. Kirsh. Today the earwig, tomorrow man? *Artificial Intelligence*, **47**, 161–184, 1991.

[34] G. F. Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press, 2001.

[35] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, **28**, 89–116, 1986.

[36] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, volume 4, B. Meltzer and D. Michie, eds, pp. 463–502. Edinburgh University Press, 1969.

[37] O. Melnik and J.B. Pollack. Theory and scope of exact representation extraction from feed-forward networks. *Cognitive Systems Research*, **3**, pp. 203–226, 2002.

[38] M. Miller. *A View of One's Past and Other Aspects of Reasoned Change in Belief*. PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 1993.

[39] M. Miller and D. Perlis. Presentations and this and that: logic in action. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society*, Boulder, Colorado, 1993.

[40] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, **13**, 103–130, 1993.

[41] T. O. Nelson. Consciousness and metacognition. *American Psychologist*, **51**, 102–16, 1996.

[42] T. O. Nelson and J. Dunlosky. Norms of paired-associate recall during multitrial learning of Swahili–English translation equivalents. *Memory*, **2**, 325–35, 1994.

[43] T. O. Nelson, J. Dunlosky, A. Graf, and L. Narens. Utilization of metacognitive judgments in the allocation of study during multitrial learning. *Psychological Science*, **4**, 207–13, 1994.

[44] M. Nirkhe, S. Kraus, M. Miller, and D. Perlis. How to (plan to) meet a deadline between *now* and *then*. *Journal of Logic and Computation*, **7**, 109–156, 1997.

[45] D. Perlis. On the consistency of commonsense reasoning. *Computational Intelligence*, **2**, 180–190, 1986.

[46] D. Perlis. Sources of, and exploiting, inconsistency: Preliminary report. *Journal of Applied Non-classical Logics*, **7**, pp. 13–24, 1997.

[47] D. Perlis, K. Purang, and C. Andersen. Conversational adequacy: mistakes are the essence. *International Journal of Human–Computer Studies*, **48**, 553–575, 1998.

[48] G. Priest. Paraconsistent logic. In *Handbook of Philosophical Logic, 2ed*, D. Gabbay and F. Guenther, eds, pp. 287–393. Kluwer Academic Publishers, 2002.

[49] G. Priest, R. Routley, and J. Norman. *Paraconsistent Logic: Essays on the Inconsistent*. Philosophia Verlag, München, 1989.

[50] K. Purang. *Systems that Detect and Repair their Own Mistakes*. PhD thesis, Department of Computer Science, University of Maryland, College Park, Maryland, 2001.

[51] K. Purang, D. Purushothaman, D. Traum, C. Andersen, D. Traum, and D. Perlis. Practical reasoning and plan execution with active logic. In *Proceedings of the IJCAI'99 Workshop on Practical Reasoning and Rationality*, 1999.

[52] N. Rescher and A. Urquhart. *Temporal Logic*. Springer-Verlag, New York, 1971.

[53] S. Russell and E. Wefald. Principles of metareasoning. *Artificial Intelligence*, **49**, 361–395, 1991.

[54] R. Sun, T. Peterson, and C. Sessions. The extraction of planning knowledge from reinforcement learning neural networks. In *Proceedings of 12th Italian Workshop on Neural Nets*, 2001.

[55] R. Sun. *Integrating Rules and Connectionism for Robust Commonsense Reasoning*. John Wiley and Sons, Inc., New York, 1994.

[56] R. Sun. Supplementing neural reinforcement learning with symbolic methods. In *Hybrid Neural Systems*, S. Wermeter and R. Sun, eds, pp. 333–47. Springer-Verlag, Berlin, 2000.

[57] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1995.

[58] D. R. Traum, C. F. Andersen, W. Chong, D. Josyula, Y. Okamoto, K. Purang, M. O'Donovan-Anderson, and D. Perlis. Representations of dialogue state for domain and task independent meta-dialogue. *Electronic Transactions on Artificial Intelligence*, **3**, 125–152, 1999.

[59] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

[60] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, **8**, 279–292, 1992.

[61] S. Wermeter and R. Sun. *Hybrid Neural Systems*. Springer-Verlag, Heidelberg, 2000.

Received 5 July 2004