

Metareasoning as an Integral Part of Commonsense and Autocognitive Reasoning

Fabrizio Morbini and Lenhart Schubert

University of Rochester

Abstract

In this paper we summarize our progress towards building a self-aware agent based on the definition of explicit self-awareness. An explicitly self-aware agent is characterized by 1) being based on an extensive and human-like knowledge base, 2) being transparent both in its behavior and in how the knowledge is represented and used, and 3) being able to communicate in natural language and directly display awareness through its dialogues. We first review the requirements imposed by explicit self-awareness on the knowledge representation and reasoning system and then describe how these have been realized in the new version of the EPILOG system. We argue that meta-level reasoning is very important for commonsense reasoning and self-awareness, but suggest in our concluding discussion that viewing agent control structure in terms of separate object-level and meta-level strata may not be particularly helpful. Rather, we suggest a “continual planning” (and execution) control structure wherein the agent’s meta-level and object-level reasoning steps mingle seamlessly, just as they do in the question-answering processes we have implemented.

Introduction

We report on progress towards building a self-aware agent based on the EPILOG inference system, an evolving implementation of the Episodic Logic (EL) knowledge representation. (Schubert 2005; Morbini & Schubert 2007) laid the foundations for our approach to self-aware agents; in summary:

- (Schubert 2005) defined explicit self-awareness (as a goal in agent design) as requiring
 - self-knowledge of human-like scope, encompassing physical, mental, autobiographical, and contextual properties;
 - encoding of self-knowledge in a form that is examinable (transparent) and usable by general inference processes; and
 - overt display of self-knowledge through communication.
- (Morbini & Schubert 2007) reported a first proof of concept of the basic ideas behind explicit self-awareness using EPILOG.

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

We have extended this preliminary work by strengthening the knowledge representations and reasoning (KR&R) capacities of EPILOG in a number of ways. The aspects most relevant to metareasoning are the handling of attitudes, substitutional quantification, and recursive self-inquiry; we discuss these in the following section. In addition we have undertaken a systematization of various facets of the inference process such as formula normalization, loop detection, multiple answer computation, and term evaluation, and discuss these in the subsequent section on inference in EPILOG. We will conclude our discussion with some subtle examples involving metaknowledge, including topical self-knowledge and time-sensitive (indexical) self-knowledge.

Metareasoning

(Schubert 2005) lists a series of requirements on the KR&R system to enable explicit self-awareness. We first summarize these requirements, and then describe in some detail how two of these requirements have been implemented in EPILOG.

- *Logic framework*: we require an explicit, transparent representation for the agent’s commonsense knowledge and metaknowledge, amenable to browsing and inferential manipulation. We chose EL as an extension of first-order logic (FOL) more fully adapted to the expressive devices of natural language (NL).
- *Events and situations*: the agent must be able to refer to events described by complex sentences (e.g., the event described by my (i.e. EPILOG’s) failure to answer a question). This capability has always been an integral part of EL and therefore of EPILOG.
- *Generic knowledge*: much of everyday knowledge is expressed using generic or probabilistic adverbs such as *usually* or *probably*. In EPILOG generics are expressed using probabilities, though this support is very limited and in need of further development.
- *Attitudes and autoepistemic inference*: the ability to reason about one’s own knowledge and to represent one’s beliefs is fundamental to self-awareness. Our commitment is to Kaplan’s computational notion of knowing (Kaplan 2000; Kaplan & Schubert 2000). We describe below how the basic machinery needed for this has been implemented in EPILOG.

- *Metasyntactic devices*: a self-aware agent needs to be able to refer in its logical language to the syntax of that language itself. How some of these devices have been implemented in EPILOG will be described later in this section.

The following subsections focus on how the last two requirements – those most central to self-awareness and metareasoning – have been implemented in EPILOG. As a notational alert, we should mention that EL uses infix notation for predication and prefix notation for function application; e.g., $(EL\ (very\ expressive))$ states that EL is very expressive; the predicate $(very\ expressive)$ is infix, while the predicate modifier $very$ (a function that maps monadic predicates to monadic predicates) is prefixed.

Substitutional Quantification and Quasi-Quotation

As motivated and exemplified in (Schubert 2005; Morbini & Schubert 2007) it is important to be able to refer to syntactic elements of EL in an EL formula itself. For instance, this allows a formal treatment of axiom schemas, enables EPILOG to classify its own predicates and formulas, and allows EPILOG to choose executable procedures for solving certain kinds of problems deliberately, based on axioms about their effects. (Some examples will be seen.) To enable this kind of “syntactic self-awareness”, EPILOG currently supports two devices: substitutional quantifiers and quasi-quotation.

Their implementation is straightforward and posed no major problems. The two main modifications required concerned the following components:

- *the parser*: we added substitutional quantifiers and metavariables. A metavariable is a particular type of variable that is bound by a substitutional quantifier. A substitutional quantifier is much like a normal quantifier except that it quantifies over substitutions of expressions of a particular category for the metavariable. For example,

$$(\forall_{wff} w (w \Rightarrow (me\ Know\ (that\ w))))$$

quantifies over substitutions of EL well-formed formulas for the variable w . The truth conditions of the formula are that all instances of the formula are true under the object-level semantics, when EL well-formed formulas (not containing metavariables) are substituted for w .¹

Quasi-quotation, written with a quote sign (apostrophe) accepts an expression as argument that may contain metavariables; substitution for such metavariables treats quasi-quotes as transparent. As an example of the use of (quasi-)quotation, we can express in EL that the “=” predicate is commutative by writing

$$(' = (Commutative\ EL\ predicate)),$$

where *Commutative* is a predicate modifier and *EL-predicate* is a predicate (true of certain syntactic objects). Other examples that will be seen later are the use of *AppearancePred* and *AppearanceFactAbout* to describe internal predicates and formulas.

- *the unification routines*: since we have provided for metavariables, unification should be able to unify not only

variables with terms but also metavariables with EL expressions of the appropriate categories. For example, we should be able to unify $(me\ Know\ (that\ w))$ with $(me\ Know\ (that\ (?x\ Foo)))$, yielding unifier $\{(?x\ Foo)/w\}$.

Recursive QA

In (Morbini & Schubert 2007) we described the basic properties a computational notion of knowing should have. We indicated why *knowing* is very different from *being able to infer*, and referred to Kaplan & Schubert’s algorithmic ASK mechanism as a basis for *knowing*. Here we describe how that notion is supported in EPILOG. The intuitive way to implement ASK (and thus to answer questions that involve predicates *Know* or *Believe*) is to allow for question-asking within a question-answering (QA) process, where the subordinate QA process is guaranteed to terminate relatively quickly. (If a question requires prolonged reasoning, the answer is by definition not known.) Answering questions about nested beliefs thus involves further nesting of QA processes. That is the basic idea behind recursive QA.

Again implementing this is straightforward in any system with a clean and modular implementation of the QA process. What is needed is that the QA process must work only on local variables and the same must be true for all systems on which QA depends (e.g., knowledge base, unification, inference, normalization, etc).

In addition to having a modular system, one needs a way to connect inference with the QA process so that this process can be started whenever it is required by some inference. In EPILOG this is achieved by using the metasyntactic devices described in the previous subsection and by providing a single special-purpose function, called “APPLY”, that the QA process knows how to evaluate whenever its arguments are quoted and metavariable-free. It executes a Lisp function of the same name for the given arguments. In particular, to implement the ASK mechanism we added the following axiom to EPILOG’s standard knowledge base; this defines “*knowing that w*” for a formula w containing no free variables as being true just in case the *knownbyme?* Lisp function returns t for argument w :

$$(\forall_{wff} w ('w\ WithoutFreeVars) ((me\ Know\ (that\ w)) \Leftrightarrow ((APPLY\ 'knownbyme?\ 'w) = 't))).$$

In effect *knownbyme?* implements the ASK mechanism as a recursive QA-process. Note that EL allows for an optional restrictor in quantified statements, and here a restrictor ($'w\ WithoutFreeVars$) is present. *WithoutFreeVars* is a predicate with one argument denoting an expression, typically specified using quotation, that is true whenever the argument contains no free variables. To evaluate this predicate EPILOG will have another axiom in its knowledge base:

$$(\forall_{subst\ x} (('x\ WithoutFreeVars) \Leftrightarrow ((APPLY\ 'withoutfreevars?\ 'x) = 't))),$$

where *withoutfreevars?* is the Lisp function that detects whether or not an EL expression contains free variables. An alternative to this approach would be to leave the attachment of procedures to EL predicates or functions implicit; however, by using explicit attachment axioms like the above, EPILOG is able to make its own reflective decisions about when to employ particular procedures.

¹Of course the omniscience claim expressed by this formula is absurd.

Currently, all formulas involving APPLY are equivalences (like the two above) in which the variables are universally quantified and usually constrained by simple conditions (like (*w WithoutFreeVars*) above). However no constraints are imposed on the syntax of formulas involving APPLY.

Whenever the QA process encounters a subgoal that contains an equality in which one of the two equated terms is an APPLY-term, where its arguments are quoted and metavariable-free, EPILOG evaluates it by executing the Lisp function specified as the first argument of APPLY, with the arguments provided for it. The result, which must be a quoted EL term, will be substituted for the APPLY term in the original equality.

Inference in EPILOG

In this section we will describe other characteristics of EPILOG's inference machinery not directly related to metareasoning but important to EPILOG's overall functioning.

Normalization

Because EL uses non-first-order constructs, e.g., substitutional quantification, quotation, lambda abstraction, and modifiers, the standard FOL normalization to clause form cannot be used. (Besides, clause form can be exponentially larger than the original form, for example for a disjunction of conjunctions.) Normalization for EL is based on term-rewriting systems, in particular on the following algorithm. (Think of EL expressions as trees, where the children of a node are the immediate subexpressions.) The two main parts of the algorithm are

1. a set of rewriting rules each divided into two parts:
 - (a) a set of preconditions. Each precondition is defined by a child identifier and by a function name. The child identifier extracts one or more descendants (subexpressions) of the EL expression currently being analysed, and the function name specifies a boolean Lisp function that takes as argument(s) the descendant(s) specified by the first part.
 - (b) a function that defines the effects of the rule. This function is executed when all preconditions are satisfied.
2. an application procedure that traverses the EL tree checking at each node if a rule applies. If a rule applies, it is executed, and if this modifies the node, control backs up N levels from the current node and the traversal is restarted from there. N is the maximum depth of the descendants used in the preconditions of the normalization rule. For example, if the rule uses only children then $N = 1$, while if some use a grandchild then $N = 2$, etc.

Currently the normalization process employs a total of 14 rules. They perform such transformations as moving negations inward, Skolemizing top-level existentials, ordering the arguments of ANDs and ORs, eliminating simple tautologies, moving quantifiers inward, etc. As in other reasoning systems, normalization contributes greatly to reasoning efficiency by collapsing classes of "obviously" equivalent formulas into unique (sets of) formulas. As an example, consider the unnormalized form of the statement, "*One email in my inbox contains no message*":

$$(\exists e0 (e0 \text{ AtAbout Now}) \\ ((\exists x ((x \text{ Email}) \text{ and } (x \text{ in MyInbox})) \\ (No y (y \text{ Message}) (x \text{ Contain } y))) ** e0)),$$

where we have simplified *my inbox* to a constant for simplicity. If this is provided to EPILOG as a fact, then normalization introduces Skolem constants for the existentials $e0$ and x , narrows the scope of the episode-characterization operator "***" to exclude atemporal conjuncts, separates implicit conjunctions, and (if we choose to include a rule mandating this) replaces $(No y \phi \psi)$ with $(\forall y \phi (not \psi))$:

$$(SK-1 \text{ AtAbout Now}), (SK-2 \text{ Email}), (SK-2 \text{ in MyInbox}) \\ ((\forall z (z \text{ Message}) (not (SK-2 \text{ Contain } z))) ** SK-1).$$

Note that normalization of a goal does not Skolemize existentials, since these serve as matchable variables. In proving a goal with a top-level universal, the universal quantifier may be eliminated and the variable given a unique new name (which thus can be thought of as the dual of Skolemization).

Inference Graph Handling

EPILOG's QA is in its simplest form a natural deduction back-chaining system. It starts with the initial question, then generates a proof subgoal and a disproof subgoal (i.e. the negation of the initial question). The QA process maintains an agenda of subgoals based on an AVL tree that decides which subgoal to process first. We have not finalized the method of sorting subgoals on the agenda; some criteria may be: subgoal complexity, their probability, their level, etc. More testing is required to decide which combination of criteria is advantageous in the majority of situations.

Each subgoal is first checked to see if it can be simplified:

- Conjunctions are split into their conjuncts, and each conjunct is handled independently of its siblings until it is solved. When a solution to a conjunct is obtained, it is properly combined with the solutions of the siblings.
- Disjunctions are split for each disjunct by assuming the negation of the remaining disjuncts. Here we make use of another feature, namely knowledge base inheritance. Each question is associated with a knowledge base that defines what knowledge can be used to answer the question. When assumptions are made, they are loaded into a new knowledge base (to be discarded at the end of the QA process) that inherits the contents of the knowledge base used before the addition of the assumptions. Currently the consistency of the assumptions is not checked, but problems will be detected from the contradictory answers produced.
- Implications, $A \Rightarrow B$, are converted into two subgoals, $(not A)$ and B assuming A .
- Equivalences are split into conjunctions.
- Universally quantified goals are simplified by generating a new constant and unifying the universal variable with it. If the universal quantifier has a restrictor, that is assumed.

When no more simplifications can be applied, goal-chaining inference is attempted. From each subgoal, one or more keys are extracted and used to retrieve knowledge. These keys are the minimal well-formed formulas embedded entirely by extensional operators such as quantifiers and

truth-functional connectives. Each subgoal maintains another agenda that decides which key to use first for retrieval. As in the case of subgoal ordering, we have not yet finalized the sorting criterion. Possibilities are preferring keys that contain more variables, or ones with the least associated knowledge.

Naturally, if a retrieved fact exactly matches a goal, then goal-chaining terminates for that goal. In the general case, goal-chaining inferences can be thought of as being resolution-like, except that the literals being resolved may be arbitrarily embedded by extensional operators. As a simple example, suppose that we have a known fact

$$(\forall x (x P) ((x Q) \text{ and } (x R))),$$

i.e., every P is a Q and an R . If we use this fact in pursuing a goal of form $((C Q) \text{ and } \phi)$ (where C is a constant and ϕ is some wff), then the derived goal will be $((C P) \text{ and } \phi)$. The extensionally embedded literals that were unified were of course $(C Q)$ in the goal and $(x Q)$ in the given fact. If one of these were intensionally embedded, for instance by reification operator “*that*”, forming an object of an attitudinal predicate, the unification and hence the goal-chaining inference would not be attempted. (For details, see, e.g., (Schubert & Hwang 2000).) For each successful inference performed for a subgoal together with a retrieved formula, a child subgoal is attached to this subgoal and the process is repeated (with termination if the derived subgoal is truth).

The two processes just described (i.e., simplification and inference) construct an inference tree. However loops and repetitions can occur, worsening performance or preventing success altogether (in case the subgoal selection behaves as a depth-first run-away). Therefore we added two optimizations, the second of which transforms the inference tree into an inference graph:

1. Loop detection: a loop is created when the same subgoal appears twice along an inference branch. In saying that a new subgoal is the “same” as a previous one, we mean that it has the same EL formula and is associated with the same knowledge base, or with a knowledge base that inherits that of the previous subgoal.
2. To avoid doing the same reasoning multiple times, we detect when the same node (where “same” is defined as above) is present on a different branch (therefore not forming a loop). In this case, we connect the two nodes and in case the already present node uses the exact same knowledge base as the new one, we completely stop further processing of the new node. If instead the new node uses a knowledge base that inherits from that of an old node we continue to process the new node as if the old didn’t exist (except for adding the connection between the two nodes).

In case the old node is answered, the answer is propagated to the new node as well.

Multiple answers

In many cases it is necessary to be able to handle multiple answers for a given subgoal, for example for wh-questions (as opposed to yes/no questions), such as “*What people do you know?*”

The main capabilities required to support multiple answers are:

- the ability to propagate the answers found (as bindings of question-variables) from the leaves of the inference graph up to the initial question. This includes taking care of merging, or waiting for the answers of sibling nodes in case they were part of a conjunction or disjunction;² and properly handling renaming of variables and the merging of unifiers.
- the ability to avoid the propagation of duplicated answers. We consider two answers the same if they use the same knowledge and produce the same unifiers.

Term Evaluation

Sometimes the answer may contain complex terms, for example functions, instead of their result as expected by whomever asked the question. For example, to the question “*How old are you?*” the system could answer with “*The difference in years between 1st of January 1993 and now*” instead of actually computing the difference.

Currently we employ a term evaluation procedure based on the same QA process. Given a complex ground term τ the question $(\exists x (x = \tau))$ is posed and the unifications for x are collected only if these are simpler than τ itself. The process is recursive, i.e., the unifications collected for x can be evaluated if they are complex terms. Because the system uses the same QA process it can detect loops and avoid duplication in the evaluation process as previously described in the subsection on inference graph handling.

However this is not ideal because the evaluation process is preprogrammed and fixed. Instead, as indicated in (Schubert 2005), we would like the QA process to automatically look for the correct type of answer as specified by *syntactic* constraints that are part of the question itself; if we ask the age of an individual, the question should probably constrain the answer to be a decimal integer providing the age in years.

Examples

In this section we describe some of the examples used to test the features of this system. We will point out how metareasoning plays a (major or minor) role in these examples.

In (Morbini & Schubert 2007) we included a preliminary discussion of the questions “*Do pigs have wings?*” and “*Did the phone ring (during some particular episode E1)?*”. Previously the examples could only be handled “socratically”, leading EPILOG through the proofs step-by-step, whereas now the questions are solved autonomously, as projected in that paper. We will not repeat the details here, but we should reiterate the claim the examples are intended to illustrate; viz., that much of our commonsense question-answering, even if not explicitly concerned with meta-level concepts, tacitly relies on metaknowledge about our own cognitive functioning. In the case of the question whether pigs have wings, we claimed that a negative answer depends on the

²In case the siblings are part of a conjunction, the propagation of the answer of any one of them must wait for a positive answer from all the siblings, since the answer must satisfy all conjuncts.

meta-belief that our “pig knowledge” is complete with respect to pigs’ major bodyparts (especially very visible ones; for contrast consider the question “*Do pigs have tonsils?*”). This *autocognitive* approach (as we termed it in (Morbini & Schubert 2007)) is not only more realistic than the usual default inference approaches, but also more efficient, because it substitutes fast ASK (self-query) checks for potentially unbounded consistency checks. Similarly the question whether the phone rang, in case of a negative answer, depends on meta-beliefs about how, and under what conditions, we acquire and retain knowledge about audible events in our environment.

One of the most interesting new questions we have tried so far is the question “*How old are you?*”. Though one can easily imagine simple short-cut methods for answering such a question, doing so in a principled, knowledge-based fashion can be non-trivial. The following table shows the knowledge used for this question:

<p>EPILOG’s birth date is 12 o’clock on the 1st of January 1993 $((date\ 1993\ 1\ 1\ 12\ 0\ 0)\ BirthDateOf\ Epilog)$</p>
<p>If x is the birth date of y then for every event e the age in years of y at the time of e is the difference in years between x and the time of e $(\forall y (\forall x (x\ (be\ (BirthDateOf\ y)))) (\forall e ((y\ HasAgeInYears\ (DiffInYears\ x\ (TimeOf\ e))) @\ e))))$</p>
<p>Time density axiom $(\forall y (\exists x (x\ AtAbout\ y)))$</p>
<p>Approximation of the meaning of AtAbout $(\forall x (\forall y (x\ AtAbout\ y) ((TimeOf\ x) = (TimeOf\ y))))$</p>
<p>Symmetry of the predicate AtAbout $(\forall x (\forall y ((x\ AtAbout\ y) \Leftrightarrow (y\ AtAbout\ x))))$</p>
<p>Axiom that says how to evaluate the predicate DiffInYears $(\forall_{term}\ x (\forall_{term}\ y ('y\ TimePointRep) (\forall_{term}\ z ('z\ TimePointRep) ((x = (DiffInYears\ y\ z)) \Leftrightarrow ('x = (APPLY\ 'diff-in-years?\ 'y\ 'z))))))$</p>
<p>Axiom that says how to evaluate the predicate TimePointRep $(\forall_{term}\ x (('x\ TimePointRep) \Leftrightarrow ((APPLY\ 'time-point-rep?\ 'x) = 't)))$</p>

The question in EL becomes

$$(\exists x (\exists e (e\ AtAbout\ Now) ((Epilog\ HasAgeInYears\ x) ** e)))$$

The answer found is that in the event (*FNSK-449 Now*) the age of EPILOG is (*DiffInYears (date 1993 1 1 12 0 0) (TimeOf (FNSK-449 Now))*) where *FNSK-449* is the Skolem function derived from the density axiom; so (*FNSK-449 Now*) identifies an event temporally near the event *Now*.

Given this answer, the evaluation of (*TimeOf (FNSK-449 Now)*) using the knowledge that (*(FNSK-449 Now) AtAbout Now*) and that

$$(\forall x (\forall y (x\ AtAbout\ y) ((TimeOf\ x) = (TimeOf\ y))))$$

produces (*TimeOf Now*), which can be evaluated to the current time.

It is in the evaluation of (*DiffInYears (date 1993 1 1 12*

0 0) (TimeOf Now)) that metareasoning comes briefly into play. EPILOG knows, in virtue of the last two axioms in the above table, that it can do the evaluation using the Lisp functions *diff-in-years?* and *time-point-rep?*. As noted before, by making EPILOG aware of the procedures at its disposal and what they accomplish (instead of leaving procedural attachment implicit), we can leave its choices of what procedural knowledge to use at what times to its own deliberate decision-making. This also opens the door to future work on learning by self-programming – the creation and purposeful use of new programs (or plans) aimed at solving specific problems.

In the current solution, as already said in the subsection on term evaluation, we explicitly call the term evaluation routine for each complex ground term returned as answer to a question.

The next question considered here is “*What is your name (now)?*”. Metareasoning enters the process only incidentally here (we’ll point out where), but to the extent that an agent’s knowledge makes reference to itself (here, through the self-referring term *Epilog*), it indicates its potential for reflective cognition. The knowledge used is displayed in the following table:

<p>Epilog-name is a name $(Epilog-name\ Name)$</p>
<p>A name is a thing $(\forall x (x\ Name) (x\ Thing))$</p>
<p>Now is during event E2 $(Now\ during\ E2)$</p>
<p>The event E2 is characterized by EPILOG having name Epilog-name $((Epilog\ Have\ Epilog-name) ** E2)$</p>
<p>Have is a continuous property: if x have y in e then x has y in all events during e. $(\forall x (\forall y (\forall e ((x\ Have\ y) ** e) (\forall z (z\ During\ e) ((x\ Have\ y) ** z))))$</p>

The question in EL is represented as

$$(\exists e0 (e0\ AtAbout\ Now) ((\exists z ((z\ Name) and\ (Epilog\ Have\ z) (\exists y (y\ Thing) (y\ (BE\ (L\ x\ (x = z)))))) ** e0))).$$

The apparently convoluted form is due to the fact that this question is automatically generated from the NL input.

After normalization we obtain the simpler question

$$(\exists e0 (e0\ AtAbout\ Now) (\exists z ((z\ Name) and\ (z\ Thing) ((Epilog\ Have\ z) ** e0))).$$

The normalization procedure moves inward the “**” operator using the knowledge that “Name” and “Thing” are atemporal predicates. This knowledge, used by the normalization procedure, is explicitly asserted in EL. This is the incidental use of metaknowledge referred to at the beginning of this example.

In the current reasoning we manually add the fact that EPILOG’s name is valid in an interval of time that includes the *Now* point. However, in future we would like this property to be automatically generated by a module in charge of maintaining the self-model of the system.

The last example shows how the metasyntactic devices

could be used to answer topical questions. The question is “What do you know about the appearance of pigs?”. The following table contains the knowledge used:

Pigs are thick-bodied. $((K (Plur Pig)) ThickBodied)$
ThickBodied is a predicate about the appearance of something. $(*ThickBodied AppearancePred)$
Every formula with structure $(x p)$ in which p is an appearance predicate is a fact about the appearance of x . $(\forall_{pred} p (*p AppearancePred))$ $(\forall x (x p) ((that (x p)) AppearanceFactAbout x)))$

The question in EL becomes

$(\exists x (x AppearanceFactAbout (K (Plur Pig))))$.

The answer found is “(that ((K (Plur Pig)) ThickBodied))”. Note that this answer depends on the metainference from the second and third axioms above that the answer wff is indeed an appearance-fact about pigs. We are not aware of any other system capable of deductive topical reasoning of this sort, in support of descriptive question-answering.

However, to retrieve more complex knowledge about pigs, for example that pigs have curly tails, more complex knowledge would have to be used.

Discussion and Further Work

The examples given show the ability of the current system to handle basic forms of metareasoning, in support of commonsense question-answering about the world and about itself. To the extent that self-modelling is crucial to self-assessment and self-improvement – the classical goals of metareasoning, it is also appropriate here to mention again our past demonstration of EPILOG’s ability to retain and make inferences from facts about its own nature, autobiography, and recent discourse events (Schubert 2005).

However, there is still much work to be done; some of the more pressing items are the following:

- Currently we have implemented only an exhaustive retrieval mechanism to be able to test the system. Given that our goal is to build an inference agent able to deal with the large knowledge base required by commonsense applications, having an efficient knowledge retrieval mechanism is crucial. Without an efficient retrieval mechanism the inference engine will only be able to answer questions using a small knowledge base.

EPILOG already had an indexing scheme designed to be scalable but some extensions are required, in particular in these directions: 1) retrieval of knowledge that uses any of the metasyntactic devices described in this paper, 2) closing some retrieval gaps, in particular in goal chaining, and 3) automatically building the type information needed to efficiently index a formula based on the most restrictive type that can be inferred for the variables in it.

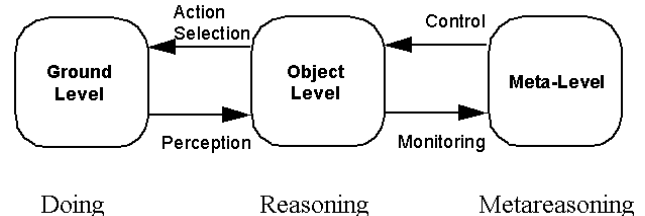
After this efficient indexing schema is finalized and implemented, we plan to test its scalability using some large knowledge base (for example, the FOL conversion of OpenCyc (Ramachandran, Reagan, & Goolsbey 2005)).

In another test we plan to assess the completeness and efficiency of retrieval and goal chaining, by running forward inference and checking how many of the inferences produced (and in how much time) can be proved by goal chaining.

- For solving many commonsense problems it seems necessary to provide the general inference engine with a set of specialized inference methods. In EPILOG these specialized reasoning routines are called specialists. In the previous version of EPILOG these specialists were “unconsciously” invoked by the general inference engine. We would like to add knowledge, based on the APPLY function, to make the inference engine aware of its specialists and their capabilities.
- Another front that needs work is the refinement of the ASK mechanism. Currently the ASK mechanism is made time-bounded simply by means of a hard limit on the depth of reasoning. (However, the limit can be computed so that several desirable properties of knowing are maintained – e.g., given that EPILOG knows A it also knows A or B).

Relationship to traditional conceptions of metareasoning

The traditional conception of the role of metareasoning in an intelligent agent is diagrammed in the following figure (Cox & Raja 2007).



The emphasis in this conception is on control and monitoring of object-level reasoning by higher-level reasoning processes, and in turn, the control of action in the world by object-level reasoning (e.g., (Cox 2005)).

It is certainly important to be clear about the distinction between procedural knowledge (e.g., executable Lisp routines for verbalizing answer formulas in English), world knowledge (e.g., knowledge about what animals have wings), and knowledge about the agent’s own internal representations (e.g., that being winged or being thick-bodied are appearance-properties of the creatures with those attributes).

From this one might infer that our agent architecture, at least at the conceptual level, fits into the diagrammed schema, with Lisp routines on the left, world knowledge in the middle, and syntactic metaknowledge on the right. However, this alignment breaks down for our architecture in a number of ways.

Most importantly, the diagrammatic schema makes meta-level reasoning the ultimate arbiter concerning the activities of the agent: meta-level reasoning controls object-level reasoning, which in turn controls physical action. We do not yet have an end-to-end self-motivated agent, only a

limited question-answering agent with some signs of self-awareness. But the overall control scheme we envisage for a purposive agent is not based on two cascaded levels of decision-making, but rather on continual modification, evaluation, and partial execution of a “life-long” plan. This plan contains hierarchically structured goals and actions (and various annotations concerning the purpose of steps, their prerequisites, effects, timing, etc.), and the unceasing pursuit of the planner is to try to improve the expected long-term net utility of the plan.

Now, certainly this plan will specify physical goals and actions (especially, for our purposes, “listening” and “speaking”). But it is also perfectly possible for goals or steps to be reasoning goals or steps (e.g., to confirm or disconfirm certain propositions, to identify tuples of entities standing in certain specified relationships, to find a hypothesis accounting for given facts, etc.) Thus, one and the same planning process can select and schedule both physical actions and reasoning actions.

It might still be argued that when it is doing the former, it is operating in object-level reasoning mode, and when it is doing the latter, it is operating in meta-level reasoning mode. However, recall our approach to accessing procedures using explicit attachment axioms that describe what a procedure accomplishes. An agent using such axioms to choose a physical action (e.g., verbalizing a formula in order to convey information) is using metaknowledge in making that choice – viz., knowledge that relates the name and I/O syntax of a procedure to the purposes it can accomplish under given conditions; thus from our perspective such reasoning is metareasoning, not object-level reasoning. Conversely, deciding on object-level reasoning goals or actions may well involve the use of object-level knowledge. For example, the decision to try to gain knowledge – a meta-level goal – by asking the user a question may hinge on whether or not the user is considered trustworthy – an object-level issue. Thus the reasoning leading to a knowledge-acquisition action certainly need not be (exclusively) a meta-level activity.

Not only does our conception of agent control blur the distinction between control of action and control of reasoning, it also blurs the distinction between “ground-level” action (doing) and reasoning. We mentioned actions such as verbalizing an answer formula in English in response to a user’s query, as a possible example of a ground-level action. However, complex cases of verbalization may involve reasoned choices “along the way”, such as choices of referring expressions, or prosodic features (e.g., stress in spoken language, or caps or italics in printed responses) that optimize communicative efficacy. Thus doing (on the left) can become inextricably entangled with object-level and even meta-level reasoning. Moreover, to the extent that an agent might encapsulate certain sequences of reasoning steps as named, executable routines if they have proved effective for certain types of problems, reasoning itself may become action-like.

So for us, the guiding picture in agent control is a continual, reward-seeking planner. The distinction between the action level, the object level, and the meta level lies primarily in the kind of knowledge that happens to be employed

in a particular reasoning or planning step. For example, one important aspect of planning is the reasoned elaboration of a step of type “achieve subgoal G” into more explicit actions to achieve G. This may well be done with the help of axioms such as one stating that “doing action(s) A brings about G”. Now, is this elaboration step an instance of object-level or meta-level reasoning? The answer depends on the syntax and semantics of A and G. If A and G refer exclusively to objects, actions and situations external to the agent’s mental contents, then the elaboration step could be regarded as an object-level reasoning step; if instead A describes a reasoning action (perhaps using attitudinal predicates such as “I try to prove that ...”, or perhaps using explicit quotation to refer to an executable inference procedure with known effects), then the elaboration step could be regarded as a meta-reasoning step. So in our conception, the distinction between meta-level and object-level reasoning is not an explicit architectural one, but rather is “hidden” in the syntactico-semantic properties of the knowledge involved. The central architectural challenge, in this view, is the formulation of the requisite planning control structure.

But then, where does that leave the issue of self-improvement through self-monitoring (the issue already mentioned above as tightly linked to metareasoning in the literature)? Given our conception of agent control, self-improvement is primarily a matter of learning to plan better. This, we think, should be a byproduct of the operation of underlying, fixed processes present from the outset. (In reasoning as in governing, the buck has to stop somewhere!) These processes include not only ones for retrieving, synthesizing and executing plans and for using available knowledge to predict consequences of contemplated actions and their net future utility, but also ones that use the episodic record of the agent’s experiences to “debug”, improve and augment the available knowledge about the world, about the agent itself, about the consequences of actions, and about their utilities to the agent. This certainly echoes the classical mantra, but we are not optimistic about its immediate and full realization. One can at present experiment with parameter adjustment as a weak form of learning (e.g., learning to prefer a given method A to a given method B under certain conditions); but we think that human-like learning presupposes human-like knowledge and human-like reasoning and planning abilities. We have chosen thus far to focus on the knowledge and reasoning infrastructure seemingly required in this formidable enterprise, and hope that what we learn will eventually shed light on how we learn.

Acknowledgements

This work was supported by NSF grant IIS-0535105 and by a 2007-2008 gift from Bosch Research and Technology Center (Palo Alto); the content has benefited significantly from the thoughtful commentary and justified prodding of the anonymous referees.

References

- Cox, M., and Raja, A. 2007. Metareasoning: A manifesto. Technical Report BBN TM-2028, BBN Technologies.

- Cox, M. 2005. Metacognition in computation: A selected research review. *Artificial Intelligence* 169(2):104–141.
- Kaplan, A. N., and Schubert, L. K. 2000. A computational model of belief. *Artif. Intell.* 120(1):119–160.
- Kaplan, A. 2000. *A Computational Model of Belief*. Ph.D. Dissertation, University of Rochester.
- Morbini, F., and Schubert, L. K. 2007. Towards realistic autocognitive inference. In *Logical Formalizations of Commonsense Reasoning*, 114–118.
- Ramachandran, D.; Reagan, P.; and Goolsbey, K. 2005. First-orderized researchcyc: Expressivity and efficiency in a common-sense ontology.
- Schubert, L., and Hwang, C. 2000. Episodic logic meets little red riding hood: A comprehensive, natural representation for language understanding. In Iwanska, L., and Shapiro, S., eds., *Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language*. Menlo Park, CA: MIT/AAAI Press. 111–174.
- Schubert, L. K. 2005. Some knowledge representation and reasoning requirements for self-awareness. In *Metacognition in Computation*, 106–113.