

Evaluation of EPILOG: a Reasoner for Episodic Logic

Fabrizio Morbini and Lenhart Schubert

University of Rochester

Abstract

It can be quite hard to objectively evaluate a reasoner geared towards commonsense problems and natural language applications if it uses a nonstandard logical language for which there exist no publicly available datasets. We describe here the evaluation of our recent improvements of the EPILOG system, a reasoner for Episodic Logic, a superset of first-order logic geared towards natural language applications. We used both a sample of interesting commonsense questions obtained from the ResearchCyc knowledge base and the standard TPTP library to provide an evaluation that tests the unique features of Episodic Logic and also puts the performance of EPILOG into perspective with respect to the state of the art in first-order logic theorem provers. The results show the extent of recent improvements to EPILOG, and that very expressive commonsense reasoners need not be grossly inefficient.

Introduction

We present here the evaluation of the progress made in the development of the EPILOG system ((Schubert et al. 1993) and (Schaeffer et al. 1993)), motivated by the recent effort towards building a self-aware agent (Morbini and Schubert 2008). EPILOG is an inference engine for Episodic Logic (EL) ((Schubert and Hwang 2000) and (Hwang and Schubert 1993)) that has been under development since 1990 ((Schubert et al. 1993) and (Schaeffer et al. 1993)).

The EPILOG system and EL are designed with natural language (NL) understanding in mind. The natural way to test its capabilities (both on the reasoning front and on the representation front) is by using a publicly available set of commonsense problems.

Among several collections that are available, we opted for the set of problems contained in the ResearchCyc knowledge base. They comprise more than 1600 problems that provide both the English formulation of a question and its translation into CycL¹. In addition to the abundance of interesting and challenging questions, another advantage of using this dataset is that it allows the comparison between our and Cyc's interpretation of each question.

The last point highlights the problem of comparison for systems that use for their evaluation a dataset based on En-

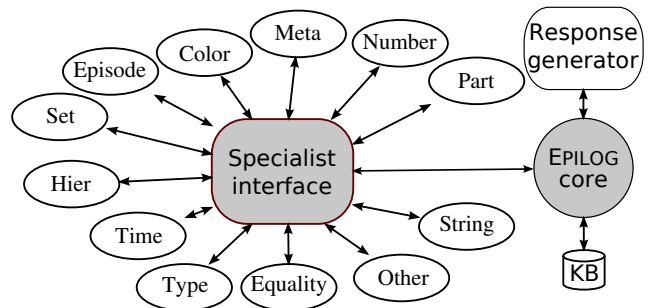


Figure 1: The high level structure of EPILOG1.

lish. Because a question expressed in English can be formalized in many ways and at various levels of detail, it is very difficult to use the results obtained to compare different systems. This lack of a dataset expressed in logic to facilitate comparisons is not easily solved given the lack of agreement on a single logical language well-suited for NL; and even if such a language existed each English sentence can still be interpreted in many ways and at different levels of detail.

Therefore, to give a more complete picture of the performance of the EPILOG system and to facilitate comparisons with other systems, we decided to evaluate it as well against the widely used TPTP dataset for FOL theorem provers. This puts the basic performance of the reasoner in perspective with respect to the state of the art in FOL theorem provers. The evaluation on Cyc's commonsense test cases instead tests the features that distinguish EPILOG from a traditional FOL theorem prover.

In the paper if we need to distinguish between the legacy EPILOG system and the new version we will refer to the former as EPILOG1 and to the latter as EPILOG2.

This paper is organized as follows: first we briefly describe the high-level structure of the EPILOG system, and then highlight the major improvements made to EPILOG in the EPILOG2 system. Then we describe in detail the evaluation of the system and state our conclusions.

EPILOG

In this section we briefly describe EPILOG and EL. Figure 1 represents the building blocks of the EPILOG1 system and

¹<http://www.cyc.com/cycdoc/ref/cycl-syntax.html>

how they are connected together. EPILOG1's core contains the inference routines, the parser, the normalizer and the storage and access schemas to retrieve and add knowledge from/to the knowledge base. A set of specialists, connected to the core inference engine through an interface module, help the general inference routines to carry out special inferences quickly (e.g., type inference to conclude whether [*Carl Artifact*] is true given that *Carl* is a coupe and coupes are a type of car, cars are vehicles, and vehicles are artifacts). The specialist interface consists of a series of flags associated with some key predicates/functions that automatically activate predefined functions in a particular specialist.

EPILOG is a reasoner for EL. EL is a highly expressive natural logic with unique features, including modifiers, reifiers, substitutional and generalized quantifiers and episodic operators, making EL particularly suited for (NL) applications. Briefly, the major differences with respect to FOL are the following.

To represent events and their relations, three *episodic operators* are introduced: *, **, and @. These operators take a well-formed formula (wff) and a term (an event) as arguments. For example, the EL formula $[[DI\ lose-control-of\ VI]\ **\ eI]$ expresses that *eI* is the event characterized by *DI* losing control of *VI*. (Note that predicates are preceded by their "subject" argument in wffs.) *Substitutional quantification* over predicative expressions, wffs, and other syntactic entities is required to express meaning postulates and introspective knowledge. It is also important for interfacing the general inference engine with specialists (as described later). *EL modifiers* correspond to the modifiers used in NL, e.g., "very", "almost" or "by sheer luck", and reification operators are used to represent generics and attitudes. *Quantifiers* allow for the use of a restrictor. For example, in the sentence "Most dogs are friendly", "dogs" is the restrictor of the quantifier "most". For the quantifiers \forall and \exists the restrictor can be incorporated into the remainder of the quantified sentence, but for many generalized quantifiers this is not possible.

EPILOG2

In this section we mention the major changes made to EPILOG1. The interface to knowledge bases (KB) has been redesigned to facilitate **1**) temporary modifications to a KB (introduced for example by the assumption-making used during inference) and **2**) the development and testing of new access schemas (i.e. mechanisms to retrieve knowledge from a KB). The result is a KB system based on inheritance of KBs (similar to what Cyc uses for inheritance of micro-theories) in which each KB is associated with a particular access schema that can be easily changed.

The parser was changed from an if-then based mechanism to a system based on a standard chart parser. This allows for easy debugging and modifications to the ever-evolving EL grammar.

The interface to specialists is now based on explicit meta-knowledge stored like any other knowledge. This knowledge specifies under what conditions a particular specialist functionality can be called. For example the formula $(\forall_{w\ f\ f}\ w\ ['w\ without-free-vars]\ [(\text{apply } 'apply-fn-knownbyme?)$

$'w) = 'yes] \Rightarrow [(that\ w)\ knownbyme])]$ describes when the introspective specialist can be called to answer whether EPILOG knows a particular formula *w*. The interface is based on this *Apply* function, which is known to the inference engine as having a special meaning.

An automatic system to extract type information has been added to EPILOG. Currently this system is used **1**) to build type hierarchies, **2**) to keep track of the return type of functions based on the type of the arguments and **3**) to build a hierarchy for the arguments of transitive predicates (also transitive predicates are automatically detected by looking for formulas like $(\forall x (\forall y (\forall z [([x\ P\ y] \wedge [y\ P\ z]) \Rightarrow [x\ P\ z]))))$, expressing transitivity).

The question-answering (QA) framework has been totally redesigned to allow for QA inside QA (used in introspection and called *recursive QA*). In addition subgoals are now selected using a hierarchical agenda that sorts the subgoals based on **1**) the size of the formula associated with subgoal *g* relative to the size of the biggest formula among the siblings of *g*; **2**) the % of times a descendant of *g* or *g* itself was selected for inference but no improvement was obtained²; **3**) the % of *g* that is solved (this is greater than 0 only for a subgoal that at some point can be split, e.g., a conjunction); **4**) the % difference between the size of *g*'s formula and the size of the smallest formula among the descendants of *g* whose solution would imply a solution of *g*; for conjunction of subgoals, their average size is considered.

Evaluation

To evaluate the progress of our effort to build a self-aware agent based on EPILOG2, we used two methods: **1**) testing on a selected small set of examples from the commonsense test cases contained in Research Cyc; **2**) the scalability test included in the TPTP library of problems for theorem provers; this scalability test was constructed from the Open-Cyc knowledge base. With the first type of evaluation we are testing the adequacy of EL for directly expressing English questions and background knowledge, and the reasoning capabilities of EPILOG2. With the second type of evaluation we are testing how EPILOG2 fares in relation to the state of the art of FOL theorem provers.

First we will describe the set of questions used to test EPILOG2's commonsense reasoning capabilities. Most of the questions have been manually encoded in EL because the general-purpose English to EL translator is not yet robust enough to handle these questions. However care has been taken not to simplify the EL form of those questions to make the job of the reasoner easier; instead we made an effort to produce EL versions that would likely be produced by an automatic, compositional English-to-EL translator. This is why some questions may appear more complex than one might expect, based on traditional "intuited" formalizations of English sentences.

In the formulas used in the following examples, we use Epi2Me as the internal constant that refers to the system itself.

²An improvement is measured either by a decrease in size of the resulting subgoal, or solution of the subgoal.

Question 1 is “How old are you?”, which in EL becomes:

$$\begin{aligned} & (\text{wh}_{term} x (\exists_{term} y [x \text{ rounds-down } y] \\ & (\exists z [y \text{ expresses } z (\text{K (plur year)})] \\ & (\exists e [e \text{ at-about Now} \\ & [[z \text{ age-of Epi2Me}] ** e]])) \end{aligned}$$

K is a reification operator that maps a predicate (here, (plur year), a predicate true of any collection of years) to a *kind* (here, the kind whose realizations are collections of years).

We have assumed that the representation of the question would be expanded pragmatically to include conventional restrictions on the form of the answer expected, i.e., an answer in rounded-down years rather than, say, seconds. These pragmatic constraints depend on the question itself; for example they would be different for a question like “How old is this bagel/star/rock/etc.?”. In the future we would like to automatically include such constraints by means of “cooperative conversation axioms”. We might have an axiom saying something like: *If X informs Y about a quantitative attribute F (such as weight, age, temperature, etc.) of some entity Z, then X is conversationally obligated to express F(Z) in units that are conventional for entities of the type(s) instantiated by Z.* In addition we would need various axioms about the conventional units for expressing weight, age, etc., of various types of entities. These axioms would then be used to refine the raw logical form of a question to include the pragmatic constraints. However, here we just focused on solving the question, manually adding the necessary pragmatic constraints.

Some of the key knowledge used to answer this question is the following:

This axiom defines the age of an entity during a particular event, when the entity’s birth date is known:

$$\begin{aligned} & (\forall y (\forall x [x \text{ (be (birth-date-of } y))]] \\ & (\forall e [[(\text{time-elapsed-between (date-of e) x) age-of y}] @ e]])) \end{aligned}$$

Axiom defining the relation between the ** and @ operators:

$$\begin{aligned} & (\forall_{wff} w (\forall e [[w @ e] \Leftrightarrow \\ & (\exists e1 [e1 \text{ same-time e}] [w ** e1]])) \end{aligned}$$

Axiom that describes which specialist function to call to express the function *time-elapsed-between* in a particular type of unit:

$$\begin{aligned} & (\forall x [x \text{ is-date}] (\forall y [y \text{ is-date}] \\ & (\forall_{pred} \text{type} [r \text{ type el-time-pred}] \\ & (\forall r [r = (\text{Apply 'diff-in-dates? 'x 'y 'type}]] \\ & [r \text{ expresses (time-elapsed-between x y)} \\ & (\text{K (plur type)}]])) \end{aligned}$$

The most interesting part of this example is the use of a set of axioms based on the *Apply* function to make the reasoning system “aware” of a set of procedures useful in computing mathematical operations and in doing type conversions. In this way EPILOG2 is able to return the answer to the question expressed as an integer that is the floor of the amount of time in years that has elapsed between the date of birth of EPILOG and now (the moment of speech). In EL the unifier found for the variable x of the initial question is: (amt 18 (K (plur year))).

Question 2 is “What’s your name?”, which expressed in EL is:

$$\begin{aligned} & (\exists e [e \text{ at-about now0}] \\ & [(\text{wh } z [[z \text{ name}] \wedge [\text{Epi2Me have } z]] \\ & (\exists y [y \text{ thing}] [y (\text{BE (L x (x = z)))]]) ** e]) \end{aligned}$$

Some of the key knowledge used to answer this question is the following:

The event now0 is during the event e2:

$$[\text{now0 during e2}]$$

The event e2 is characterized by EPILOG having the name ‘epilog-name’:

$$[[\text{Epi2Me have 'epilog-name}] ** e2]$$

If one event is characterized by something possessing something else, then that will also be true for any event during the first event:

$$\begin{aligned} & (\forall x (\forall y (\forall z [[x \text{ have } y] ** z] \\ & (\forall zz [zz \text{ during } z] [[x \text{ have } y] @ zz]])) \end{aligned}$$

Of interest here is the last axiom because it ascribes “inward persistence” (homogeneity) to predicate *have*, a property it shares with other *atelic* predicates. The two other formulas are hand-additions to the current knowledge base, but they should be automatically inserted, the first by the English to EL generator, the second by a self-awareness demon that is in charge of maintaining basic information about the agent, for instance, its name, its state (e.g. sleeping, awake, etc.) and its “state of health” (e.g., cpu consumption, free memory, garbage collection status, etc.).

To correctly answer this question the reasoner also uses lexical knowledge that states which predicates are atemporal and therefore can be moved out of the scope of the ** operator. This knowledge is expressed in EL and it is used by the normalizer. An example is (‘thing EL-type-pred), stating that ‘thing’ is a type predicate and therefore atemporal.

Question 3 shows how EPILOG could answer questions about its own knowledge. The question is “What do you know about the appearance of pigs?”, which in EL we expressed as:

$$(\text{wh } x [x \text{ appearance-fact-about (K (plur pig))}]$$

Some of the relevant knowledge involved in this example is:

Pigs are thick-bodied:

$$[(\text{K (plur pig)}) \text{ thick-bodied}]$$

The predicate ‘thick-bodied’ is an appearance predicate:

$$[r \text{ thick-bodied appearance-pred}]$$

Every wff that uses an appearance predicate is a fact about the appearance of its subject:

$$\begin{aligned} & (\forall_{pred} p [r \text{ p appearance-pred}] \\ & (\forall x [x p] [(that [x p]) \text{ appearance-fact-about } x]) \end{aligned}$$

One could construct much more complex formulas pertaining to the appearance of something, e.g., that the appearance of a person’s hair – say, color and style – constitutes appearance information about the person.

The remaining questions are taken from the ResearchCyc 1.0 collection of commonsense test cases. About 81% of these test cases have been axiomatized to become solvable

by Cyc; among those presented here, the last two have a solution in Cyc. An important difference between our and Cyc's approach to these problems is in the style of formalization: Cyc's representations are in a simplified form that **1**) is geared towards the CycL style (e.g., using many concatenated names for complex expressions instead of compositionally combining the parts), which is far from NL-based representations; and **2**) omits important details (e.g. temporal relations) and pragmatic constraints.

Question 4 is “*Can gasoline be used to put out a fire?*”. In Cyc this is the test case named # $\$CST$ -Can-YouUseGasToPutOutAFire, and the question is expressed as: ((TypeCapableFn behavior-Capable) GasolineFuel ExtinguishingA-Fire instrument-Generic). (TypeCapableFn behaviorCapable) returns a predicate that describes the capacity for a certain behavior of a certain type of thing in a certain role position. In effect the question becomes, “Is gasoline-fuel behaviorally-capable of being a generic-instrument in fire-extinguishing?”

We also interpret the question generically, but we adhere more closely to a possible English phrasing, asking whether there could be an instance where a person uses gasoline to put out a fire:

$$\begin{aligned} &(\exists e [e \text{ during (extended-present-rel-to Now)}] \\ &(\exists x [x \text{ person}] \\ &(\exists y [y \text{ ((nn gasoline) fuel)}] \\ &(\exists z [z \text{ fire}] \\ &[[x \text{ (able-to ((in-order-to (put-out z)) (use y))))} \\ &@ e]]))) \end{aligned}$$

Some of the knowledge relevant to this question is:

If some person is able to use some stuff to put-out a fire then s/he must be at the same location as the fire, must have at hand that stuff and that stuff must be flame-suppressant:

$$\begin{aligned} &(\forall e [e \text{ during (extended-present-rel-to Now)}] \\ &(\forall x [x \text{ person}] (\forall y [y \text{ stuff}] (\forall z [z \text{ fire}] \\ &([x \text{ (able-to ((in-order-to (put-out z)) (use y))))} @ e] \\ &\Rightarrow [[x \text{ has-at-hand y} @ e] \wedge [x \text{ loc-at z} @ e] \\ &[y \text{ flame-suppressant}]]]]]]) \end{aligned}$$

Gasoline is flammable stuff:

$$(\forall x [x \text{ ((nn gasoline) fuel)}] [[x \text{ flammable}] \wedge [x \text{ stuff}]])$$

Flammable things are not flame-suppressant:

$$(\forall x [x \text{ flammable}] (\text{not } [x \text{ flame-suppressant}]))$$

The question is answered negatively by using the knowledge that to be able to put-out a fire one must use a flame-suppressant material, and gasoline is not a flame-suppressant material.

Question 5 is Cyc's question named # $\$CST$ -DoesCyc-HaveABiologicalFather, which in English is “*Do you (Cyc) have a biological father?*”. In Cyc the question is represented as (thereExists ?F (biological-Father Cyc ?F)).

We expressed the question in EL as follows:

$$\begin{aligned} &(\exists e [e \text{ at-about Now}] \\ &(\exists y [[Epi2Me \text{ (have-as ((attr biological) father)) y} \\ & ** e]]) \end{aligned}$$

In this question, *have-as* is a so-called “subject-adding operator” that takes a unary predicate as argument and returns a binary predicate. In this case ((attr biological) father) is the monadic predicate true for all individuals that are biological fathers. (have-as ((attr biological) father)) is the binary predicate that is true for all pairs of individuals in which the object of the predicate is the father of its subject.

The relevant knowledge for this example is:

EPILOG is an artifact:

$$[Epi2Me \text{ artifact}]$$

No artifact is a natural object:

$$(\forall x [x \text{ artifact}] (\text{not } [x \text{ natural-obj}]])$$

A creature is a natural object:

$$(\forall x [x \text{ creature}] [x \text{ natural-obj}])$$

All creatures have a biological father:

$$\begin{aligned} &(\forall x [[x \text{ creature}] \Leftrightarrow \\ &(\exists y (\exists e \\ &[[x \text{ (have-as ((attr biological) father)) y} ** e]])]) \end{aligned}$$

The question is answered negatively by using the knowledge that EPILOG is an artificial thing and therefore not a natural object. Further it is known that only creatures can have a biological father and that creatures are a subtype of natural objects.

Question 6 corresponds to Cyc's question named # $\$CST$ -AnimalsDontHaveFruitAsAnatomical-Parts-HypothesizedQueryTest In Cyc the question is expressed as (implies (isa ?ANIMAL Animal) (not (relationInstanceExists anatomicalParts ?ANIMAL Fruit))).

In EL we express the question (more naturally, we claim) as:

$$\begin{aligned} &(\forall e [e \text{ during (extended-present-rel-to Now)}] \\ &(\text{No } x [x \text{ animal}] \\ &[[x \text{ (have-as anatomical-part) (K fruit)}] ** e]]) \end{aligned}$$

The function *extended-present-rel-to* applied to an event *e* returns the event that started long ago and continues long pass the end of the event *e*. The extent of the event returned should be context-dependent. However, for this question this is irrelevant given that the knowledge used is presumed true for any event. The relevant knowledge for this example is:

Plant stuff is not animal stuff:

$$(\forall x [x \text{ plant-stuff}] (\text{not } [x \text{ animal-stuff}]])$$

Fruits are made of plant stuff:

$$[(K \text{ fruit}) \text{ made-of } (K \text{ plant-stuff})]$$

Animals are made of animal stuff:

$$[(K \text{ animal}) \text{ made-of } (K \text{ animal-stuff})]$$

If an individual *x* is made of (kind of stuff) *p* and if (kind of stuff) *q* is a subtype of *p* then *x* is made of *q*:

$$\begin{aligned} &(\forall x (\forall_{pred} p [x \text{ made-of } (k p)] \\ &(\forall_{pred} q (\forall y [y p] [y q] \\ &[x \text{ made-of } (k q)])) \end{aligned}$$

If an individual *x* is made of (kind of stuff) *p* and if (kind of stuff) *q* is disjoint from *p* then *x* is not made of *q*:

Segment	Size (min/avg/max)	EPILOG1 FI	EPILOG1 no FI	EPILOG2	Avg depth	Vampire 9
1	(22/59/163)	46	46	100	5.9	100
2	(-/1101/-)	46	44	92	5.6	100
3	(-/7294/-)	0	0	54	4.5	82
4	(-/42981/-)	0	0	48	4.3	32
5	(-/534435/-)	0	0	12	1.3	0

Table 1: Summary of the tests carried out between EPILOG1, EPILOG2 and the Vampire theorem prover, version 9. The first column contains the segment number (1-5) of the segments comprising the scalability subset of the CSR dataset (with 50 problems in each segment). Column 2 lists min, max and average number of formulas contained in the problems in that specific segment. (If all problems contain the same number of formulas only the average is shown). Columns 3, 4, and 5 show the percentage of problems for which a solution was found, respectively by EPILOG1 with forward inference enabled, EPILOG1 without forward inference and EPILOG2 (which by default has no forward inference enabled). Column 6 shows the average depth of the answer found by EPILOG2. Column 7 shows the percentage of problems solved by Vampire. All systems have been limited to a timeout of 120 seconds.

tative of state-of-the-art FOL theorem provers⁸. All systems were run under the same conditions and were subjected to a 2 minute limit per problem.

Conclusion and Further Work

In this paper we described how we evaluated the work on the development of the latest version of the EPILOG system in a way that we think tests the particular features that characterize EPILOG and that also may allow for comparison with other commonsense reasoners independently of which logical language they use.⁹

The evaluation was divided into 2 parts. In the first we selected 8 examples, five of which were from ResearchCyc. These examples were selected to test the features of EL and of EPILOG such as introspective question answering, quotation and substitutional quantification, interfacing to specialists, etc. The second part was based on a subset of the TPTP dataset used to test the scalability of a theorem prover. This part, in addition to providing a baseline for assessing future enhancements of EPILOG, demonstrates significant performance gains achieved here over EPILOG1, and will facilitate further comparisons with other theorem provers. Moreover, the results show that a reasoner for a highly expressive logic doesn't have to be impractically inefficient compared to a less expressive one¹⁰. It should be kept in mind that in addition to not lagging far behind state-of-the-art performance in FOL theorem provers in their domain of competence, EPILOG is capable of additional modes of reasoning and metareasoning as shown by the first evaluation.

In future we plan to close the remaining gap between EPILOG and FOL theorem provers, implement a more efficient access schema for knowledge retrieval, implement probabilistic reasoning, provide for uniform handling of generalized quantifiers, and extend the new approach to specialist deployment to all specialists.

⁸Download available at <http://www.cs.miami.edu/~tptp/CASC/J4/Systems.tgz>

⁹Allowing longer times had minimal effect on both systems.

¹⁰contrary to the alleged "expressivity/tractability tradeoff".

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 very useful comments of the anonymous referees.

References

- Hwang, C., and Schubert, L. 1993. Episodic logic: A situational logic for natural language processing. In P. Aczel, D. Israel, Y. K., and Peters, S., eds., *Situation Theory and its Applications*, volume 3. Stanford, CA: Center for the Study of Language and Information. 303–338.
- Morbini, F., and Schubert, L. K. 2008. Metareasoning as an integral part of commonsense and autocognitive reasoning. In *Metareasoning 08*, 155–162.
- Ramachandran, D.; Reagan, P.; and Goolsbey, K. 2005. First-Orderized ResearchCyc: Expressivity and Efficiency in a Common-Sense Ontology.
- Schaeffer, S.; Hwang, C.; de Haan, J.; and Schubert, L. 1993. EPILOG, the computational system for episodic logic: User's guide. Technical report, Dept. of Computing Science, Univ. of Alberta.
- 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.; Schaeffer, S.; Hwang, C. H.; and de Haan, J. 1993. *EPILOG: The Computational System for Episodic Logic. USER GUIDE*.