Knowledge Representation Integrating Conditional Probabilities, Closure Logic, and Primitives

Robert L. Kirby, Ph.D.
USA
kirby.bob@gmail.com

Abstract

Work in progress implementing a knowledge representation (KR) integrating estimates of conditional probabilities with propositions from closure logic whose predicates are governed by a small set of primitives. The KR aspires to support Artificial General Intelligence (AGI) by representing anything an AGI needs and giving appropriate access including queries. An initial demonstration will parse text with word meanings and natural language syntax encoded in the KR to store semantics.

1 AGI support

Fully general Artificial Intelligence should handle any information. Traditional implementations had an explicit knowledge representation [Brachman and Levesque, 2004], which creates, reads, updates, and deletes (CRUD) data. The reads include queries that take a description of a situation and return results to surrounding layers. Having an explicit KR can allow better understanding of the workings of the entire system. However, a KR itself need not provide user friendly explanations or extensive computations but instead, layers using the KR can do that. The requirements of a KR include the ability to represent anything including contexts, uncertainty, and other graded information; adaptability to new information; and reasonably efficient CRUD operations. The goals of a KR should support a normatively helpful system, not necessarily one that behaves like a human; support understandability through explanations; use reliable technologies, presumably based on principled, well-understood, robust disciplines; be parsimonious with system details and resource usage; and potentially secure by being continuously available but providing only authorized access.
The approach given below that unifies a limited set of predicates in closure logic statement in conditional probabilities is hypothesized to satisfy the KR requirements and goals.
If natural language with its denotations, connotations, and syntax is enough to represent anything, then a smaller set of vocabulary and relationships could also be enough but also allow efficient computation. But how small? Gottfried Wilhelm Leibniz postulated that a limited system, with a few thousand parts, a *characteristica universalis*, could be sufficient for computation. The Stanford MARGIE project [Rieger, 1974], which Roger Shank led, used a small quantity (<50) of names and relations to represent many commonplace ideas within limited resources. Current explorations incorporating names and relations from various sources have shown that less than a hundred primitives, some of which refer to constants from unbounded set of strings, rational numbers, and other limited sets of names such as named irrational numbers seem to be enough to represent anything so far. Such sets of primitives with intentional strict, unambiguous definitions that overlap as little as possible may be combined to represent much, if not all, of the diverse information that a KR should include.
Closure logic provides a way to efficiently combine primitives, adapt to new information, and support access from the modules or layers of an AGI. Closure logic supports meta statements about statements, which natural language modal auxilaries and modal logic represent. A KR using closure logic can represent such meta statements and other constructs such as metaphors and paradoxes, but does not internally support more complicated inferences about them, which is outside of the requirements of a KR. Closure logic can represent graph structures, which have been proposed for other KRs and syntax, including current efforts to represent a substantial part of American English natural language syntax.
Closure logic combining primitives may be enough to represent anything, including probabilities. However, explicit condtional probabilities whose propositions are closure logic statements improve efficiency, help statisfy KR goals, and provide appropriate query mechanisms. The conditions of conditional probabilities describe situations for which queries may return results, the conditionees of the query condition and its more generic conditions. The caller gets probabilities to evaluate results and decide future processing. Backtraces and logs can fully explain processing decisions and their results. With syntax encoded as conditional probabilities, parsing even non-standard natural language, while taking advantage of word semantics, becomes possible. Merging conditionees of queries into more specific statements may summarize natural language, answer questions, identify relevant information in large corpori, participate in dialogs, support principled decisions,

or execute commands. More details about how such a KR may support natural language understanding and other applications of AGI follow.

2 Conditional Probabilities

The conditional probability estimates judge the discrete probability of a conditionee event when its condition event occurs from an ideal distribution. (`0≤estimate≤1`) If the condition `B`, a proposition, of a conditional probability `P(A|B)=p` is true then the conditionee `A`, another proposition that implies `B`, can be interpreted to occur with likelihood p. The propositions `A` and `B` model events, which are sets of outcomes, which are individual situations, real or imagined, that the propositions distinguish. The estimate may be gotten from a model, such as a probability distribution, statistics, or a human a priori evaluation. Implementation properties of the condition or conditionee could include history or other provenance along with other useful information. Some conditional probability groupings could even describe the probability of a probability, called a second-order probability [Good, 1965].

Conditional probabilities should include enough details so that all assumptions are explicit. Conditional probabilities are similar to the if-then rules of expert systems except that the consequence of expert system rules is often written to be executed [Friedman-Hill, 2003], while a caller would evaluate the corresponding conditionee. Like expert systems, conditional probabilities can provide powerful explanations and guide complex decisions.

Probabilities, along with a schema for assigning values or worth, could provide utilities for making decisions, both internal to implementation processing and external. There are formulas for combining conditional probability estimates and fit measures to create new conditional probability estimates with a linear amount of computation. A system could use such formulas to refine estimates as more context is added about a statement, such as during parsing, mentioned below.

The conditions of conditional probabilities form a transitive reduction directed acyclic graph (DAG) with generalization defining edges. This DAG is implemented as both temporary and persistent stores, which support CRUD operations. Lookup of conditions, which may include any generalized conditions, provides default reasoning, with the conditional probabilities of the most relevant conditions taking precedence. Tied condition relevance may be adjudicated numerically, sometimes considering the conditional probabilities of less relevant conditions. This approach obviates any need for nonmonotonic reasoning.

The DAG provides the information of an ontology, with more specific conditions, which imply other conditions, corresponding to nodes close to the leaves of an ontology. The upper levels of ontologies, which have few or no constraints, may not be needed. Rather than attaching properties to the nodes of an ontology, DAG conditions may be more precisely specified, leaving out irrelevant constraints that might tend to be added when constructing an ontology, which may lump all constraints of each category together.

Callers may use the DAG incrementally, working first with more generic conditions, which have fewer constraints, to perform rough evaluation and then selecting more specific conditions, which may involve more expensive computations. For example, during initial parsing of natural language, coarse meanings of words and potential syntax might be combined in several ways in order to set aside poorer alter alternatives and explore better ones.

For specific applications, the DAG may be bounded to a few domains, possibly fitting on a cell phone, but still allowing relevant information to be inserted. Even with a larger DAG, only relevant information may be retrieved without losing efficiency.

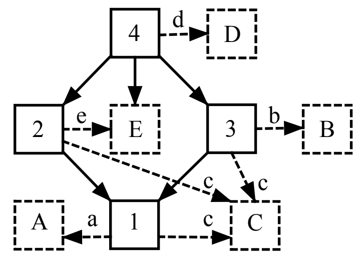

Figure 1: Conditions DAG

Figure 1 illustrates conditional probability visibility within the transitive reduction DAG of conditions. Solid lines indicate conditions and their generalizations. Dashed lines connect conditions to their conditionee events with a probability estimate annotation. Conditions 2 and 3 with more constraints are more specific than condition 1. Condition 4 includes constraints from conditionee event E and conditions 2 and 3. The event A estimate of condition 1 is the default for the other conditions. The event B estimate of condition 3 is only the default for condition 4. Event C estimates of conditions 1, 2, and 3 may combine for an estimate for condition 4. The event D estimate is only visible to condition 4. The event E estimate of condition 2 is irrelevant to condition 4 since 4 incorporates E.

3 Closure logic

Closure logic expressions specify the propositions of conditional probabilities with conjoined constraints. Its constraints each have a primitive name; a closure (a neutrally named programming concept); a constant, which may be null; and at least one variable or entity parameter as its primitive allows. Parameters are considered as variables in logic and as entities for modeling.

Closure logic restricts IKRIS [IKRIS, 2018] logic (ICL), which formalized the diagrams of John Sowa [Sowa, 2000]. ICL is similar to the eventualities of Hobbs [Gordon and Hobbs, 2017]. ICL may correspond to the context logic using first order logic (FOL) that Ohlbach investigated [Ohlbach 1989]. ICL does not quantify over predicates like higher-order logic (HOL).

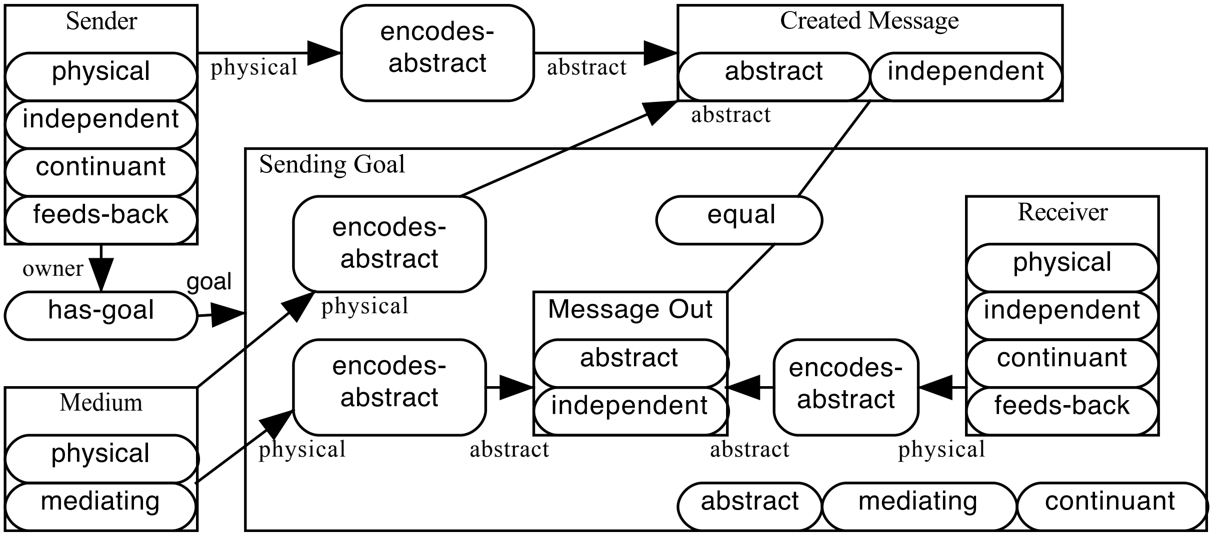

Figure 2: Message goal part of word definition

Closure logic has closures, which correspond to the containing boxes in the diagrams of John Sowa. To allow a consistent implementation, every constraint has a closure, which for most constraints will be in the outermost closure, which is aligned with an entire proposition, either a condition or conditionee. As used here, closure logic allows a singly rooted DAG of closures, with predicates, which are each contained in some closure including a global closure, to have some arguments that are also closures. Closures may assume many different roles such as a context, situation, state, eventuality, goal, desire, plan, proposal, thought, imagining, belief, suspicion, possibility, hypothesis, or process. Within a closure, all constraints are conjoined, ANDed together.

An important special primitive is classical negation (¬), which has a single parameter, which is a closure, and a null constant. With negation, DeMorgan's laws allow the creation of expressions with equivalents to quantifications (∀, ∃), disjunction (∨), material implication (→), and other connectives of FOL.

Closures provide equivalent expressions such as the targets of modal auxilaries like “will”, “may”, “can”, “must”, “need”, and “dare” that modal logic could express. Metaphors [Way, 1991] and other analogies, which involve situations where some constraints are not literally true, may be represented in a closure. Even paradoxes may be represented in a closure.

Figure 2 is a portion giving the goal parts of expressions defining words, omitting process. A sender has a goal to have a message, which a medium carries, received with the same content. Rounded boxes show predicates, with the only strings that computations use: their primitive name, optional constant, or property identifier. Other annotations improve human readability. Rectangular boxes show entities, which contain their properties, which are typically distinctions that Charles Sanders Pierce described. The goal is the only non-global closure; the global closure is implicit. Arrows first into and later outward indicate arguments of constraints that are not simple properties.

4 Primitives

Primitives, which are a quite limited set of predicate groupings, specify all constraints in closure logic expressions. Corresponding to the constraints of expressions, primitives each have a name, an ASCII string; a type allowed for constraint constants; and parameter specification. The constant types currently considered are numbers and strings although other types such a groupings for points or bounding boxes may be appropriate.

The parameter specification states the minimum quantity, which is also the maximum for most parameters and currently three (3) or less. In order to deal with permutations, some primitives may specify that all parameters after a specified parameter are interchangeable, allowing a single constraint in place of multiple constraints, which would only vary in the order of their arguments. A few primitives, like an abstract `tuple`, may allow a varying quantity of arguments, where a particular constraint is only considered the same as another constraint if it has the same quantity of arguments. Other primitives, like `less-than`, allow a varying argument quantity without a requirement for quantities to match.

The features of primitives were chosen to facilitate implementation. The choice of primitives is not unique but conversely has not needed much change. There were several inspirations for primitives. John Sowa [Sowa, 2000] suggested many primitives based on the work of Charles Sanders Pierce that are similar to the distinctions of upper-level ontologies. The Stanford MARGIE project published many relations, as names and symbols, which led to primitives, including scales of human state, such as pain and attention, which might help describe sentiments and connotations. Some MARGIE relations, like PTRANS, were reorganized into orthogonal or more refined parts, such a process with a before and after state. Natural Semantic Metalanguage (NSM) of Anna Wierzbicka, which Cliff Goddard extends, provides another view of common ideas that primitives should include. A few other primitives came from considering math and science, such as measurements. Further primitives may be needed but their inclusion may become rarer and rarer. Perhaps a goal of producing the *characteristica universalis* of Leibniz may be reached.

5 Modules and Layers

What the KR does independently is limited. Applications, modules, and layers access the KR to do tasks. An important module heuristically matches variables or entities of closure logic expressions. Its algorithm insures that matched entity properties are consistent. For instance, a matched entity may not be both physical and abstract. Previous expression matches, such as of conditions, may constrain further entity matches, such as those of their conditionees. The formulas for combining conditional probability estimates can incorporate the goodness of fit of a match. If the probability of a combination or its combination with other combinations is low, the caller may restart the matching algorithm to identify the next best fit. With a limited size set of primitives, which have well understood meanings applied consistently to form expressions, the probability estimates are more meaningful.

Natural language understanding matches word semantic expressions and syntax encoded as closure logic expressions rather than a traditional grammar parsing approach. Word meaning, semantics, and natural language syntax statements can combine conditional probabilities to guide stochastic parsing. Lower probability syntax statements can allow robust parsing including ungrammatical inputs. The most likely combinations of semantics and syntax are filtered to produce larger and larger combinations while allowing nearly equivalent likelihoods to compete or be summarized. When parsing combinations of larger portions of input, such as clauses, the matching algorithm can identify repeated references to an entity, addressing linguistic coreference issues.

Other software layers can add capabilities, such as calling parsing as part of satisfying tasks. A long-running task could request inputs, add declarative information to a store, and potentially execute interrogative or imperative statements for an interlocutor with sufficient permission, the function of a chatbot. Another task could read formatted linguistic repositories, such as Wordnet [Fellbaum, 2005], extract parts of entries and then parse any natural language to acquire meanings for new words using already understood meanings. A narration task could parse a story, extract a summary from the primary actions, and organize the details to answer questions or support searches.

6 Plans and Resources

A single independent researcher with help limited to evaluation and advice is developing a text demonstration. An existing website, `http://bobkirby.info/`, supports registered users curating primitives and conditional probability statements and may host initial demonstrations, like a chatbot. Eventually, the vocabulary will need to be expanded to include perhaps the most frequent words of American English or those of problem sets, such as the Winograd Challenge [Winograd, 2018]. A flushed-out demonstration should spark imagining even more possibilities.

Acknowledgments

Prof. Charles Rieger III, Alan Terry, and Steve Greidinger provided advice and encouragement. Dr. Mary Usher allowed persistence. For Rose.

References

[Brachman and Levesque, 2004] Ronald J. Brachman and Hector J. Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann, Elsevier 2004

[Fellbaum, 2005] Christiane Fellbaum. WordNet and wordnets. In Brown, Keith et al. (eds.), *Encyclopedia of Language and Linguistics*, Second Edition, Oxford, Elsevier, 2005.

[Good, 1965] Irving John Good. *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*. Research Monograph No. 30, The M.I.T. Press, Cambridge, Massachusetts, 1965.

[Gordon and Hobbs, 2017] Andrew S. Gordon and Jerry R. Hobbs. Eventualities and Their Structure. In *A Formal Theory of Commonsense Psychology: How People Think People Think*. pp 93-99 Cambridge University Press, 2017.

[Friedman-Hill, 2003] Ernest Friedman-Hill. *Rule-Based Systems in Java*. Manning Publications, June 2003.

[IKRIS, 2018] `http://www.ihmc.us/users/phayes/IKL/GUIDE/GUIDE.htm`

[Ohlbach 1989] Hans Jürgen Ohlbach. *Context Logic*. SEKI-report, SR–89–08, FB. Informatik, University of Kai-serslautern, 1989.

[Rieger, 1974] Charles J. Rieger, III. *Conceptual Memory: A Theory and Computer Program for Processing the Meaning Content of Natural Language Utterances*. Stanford Artificial Intelligence Laboratory Memo AIM-233, STAN-CS-74-419, 1974.

[Sowa, 2000] John F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[Way, 1991] Eileen Cornell Way. *Knowledge Representation and Metaphor*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.

[Winograd, 2018] Winograd Schema Challenge. http://commonsensereasoning.org/winograd.html, 2018.

