Acquiring and Using World Knowledge using a Restricted Subset of English

Peter Clark, Phil Harrison, Tom Jenkins, John Thompson, Rick Wojcik
Mathematics and Computing Technology
Boeing Phantom Works
P.O. Box 3707, Seattle, WA 98124
{peter.e.clark,philip.harrison,thomas.l.jenkins2,john.a.thompson,richard.h.wojcik}@boeing.com

Abstract

Many AI applications require a base of world knowledge to support reasoning. However, construction of such inference-capable knowledge bases, even if constrained in coverage, remains one of the major challenges of AI. Authoring knowledge in formal logic is too complex a task for many users, while knowledge authored in unconstrained natural language is generally too difficult for computers to understand. However, there is an intermediate position, which we are pursuing, namely authoring knowledge in a restricted subset of natural language. Our claim is that this approach hits a "sweet spot" between the former two extremes, being both usable by humans and understandable by machines. We have developed such a language (called CPL, Computer-Processable Language), an interpreter, and a reasoner, and have used them to encode approximately 1000 "commonsense" rules (a mixture of general and domain-specific). The knowledge base is being used experimentally for semantic retrieval of video clips based on their captions, also expressed in CPL. In this paper, we describe CPL, its interpretation, and its use for reasoning, and discuss the strengths and weaknesses of restricted natural language as a the basis for knowledge representation.

Introduction

Despite many recent advances, knowledge acquisition remains a major bottleneck in AI. While automated methods are effective for some classes of knowledge, manual methods are still needed when applications require deeper, axiom-based representations for the tasks they perform. However, authoring knowledge in formal logic is too complex a task for many users, while knowledge authored in unconstrained natural language is generally too difficult for computers to understand. Nevertheless, there is an intermediate position, which we are pursuing, namely authoring knowledge in a restricted subset of natural language. We have developed such a language (called CPL, Computer-Processable Language), an interpreter, and a reasoner, and have used it to encode approximately 1000 "commonsense" rules. In this paper, we describe CPL, its interpretation, and its use for reasoning, and discuss the strengths and weaknesses of restricted natural language as a the basis for knowledge acquisition and representation.

While there has been a substantial amount of prior work on Controlled Language (CL) processing, the majority of it has been devoted to making text easier for people to understand, e.g., (Hoard, Wojcik, & Holzhauser 1992; Mitamura *et al.* 2003), rather than, as is our goal here, to make text easier for computers to understand (we will refer to this as a "computer-processable" rather than "controlled" language). Despite this, there are several ongoing projects with computer-processable languages that have demonstrated the viability of the approach, e.g., (Fuchs, Schwertel, & Schwitter 1998; Schwitter & Tilbrook 2004), and the commercial success of some controlled languages, e.g., AECMA simplified English, suggests that people can indeed learn to work with restricted English. This has inspired us to pursue this avenue, extending the expressiveness and inference capabilities of these earlier systems.

Using restricted English offers several advantages:

1. It seems to be easier and faster than encoding knowledge in a formal KR language directly, although (as we discuss later) we consider it more restricted
2. It is more comprehensible and accessible to typical users
3. It is ontologically less commital: If a design decision concerning the target ontology is changed, the original input sentences can be automatically reprocessed using the modified language interpreter.
4. It is a step towards richer language processing

Despite this, there are several significant challenges in using a computer-processable language, in particular it still requires some skill to harness the language effectively. We first describe our language CPL, and then discuss these issues in detail.

CPL: A Computer-Processable Language

The CPL Language

A basic CPL sentence has the structure:

subject + verb + complements + adjuncts

where complements are obligatory elements required to complete the sentence, and adjuncts are optional modifiers. Within this structure, the CPL grammar currently includes handling of prepositional phrases, compound nouns, ordinal modifiers, proper nouns, adjectives, passive and gerundive forms, relative clauses, and limited handling of conjunctive coordination. Pronouns are not allowed; instead, the user must use definite reference. Basic sentences can be conjoined together using the keyword "AND". Figure 1 shows some example CPL sentences and rules.

As we describe shortly, basic sentences are interpreted as ground "facts" about the world, i.e., objects are considered existentially quantified. To enter rules of inference, i.e., statements involving universal quantification, CPL uses seven rule templates, whose elements are basic CPL sentences. One such template is:

(1) “A man picks up a large box from a table”
(2) “The man carries the box across the room”
(3) “The man is sweeping the powder with a broom”
(4) “Two vehicles drive past the factory’s hangar doors”
(5) “The narrator is walking past racks of equipment”
(6) “The narrator is standing beside a railing beside a stormwater outfall”

(7) “IF a person is carrying an entity that is inside a room THEN (almost) always the person is in the room.”
(8) “IF a person is picking an object up THEN (almost) always the person is holding the object.”
(9) “IF an entity is near a 2nd entity AND the 2nd entity contains a 3rd entity THEN usually the 1st entity is near the 3rd entity.”
(10) “ABOUT boxes: usually a box has a lid.”
(11) “BEFORE a person gives an object, (almost) always the person possesses the object.”
(12) “AFTER a person closes a barrier, (almost) always the barrier is shut.”

Figure 1: Example sentences (1)-(6) and rules (7)-(12) expressed in CPL.

If *sentence1* then *typically sentence2*

where *sentence1* and *sentence2* are basic CPL sentences, and *typically* is a qualitative degree of reliability. The Web-based interface for entering rules is shown in Figure 2. We describe the rule templates and their interpretation later, after first describing the basic processing of a CPL sentence.

The Target KR Language

The goal of CPL interpretation is to translate the original English sentences into a formal knowledge representation (KR) with well-defined semantics, which can then be used in reasoning and question-answering. The target KR language we are using is called KM, the Knowledge Machine (Clark & Porter 1999). KM is a mature, advanced, frame-based language with well-defined semantics, used previously in several major KR projects.

Interpreting CPL

We now describe the technical details of how basic CPL sentences are interpreted. Following this, we discuss how sentences involving universal quantification are processed.

CPL sentences are translated to logic in three main steps, namely parsing, generation of an intermediate “logical form” (LF), and conversion of the LF to statements in the KM knowledge representation language. Each step exploits semantic information to guide the interpretation.

Parsing Parsing is performed using SAPIR, a mature, bottom-up, broad coverage chart parser, also used commercially (Harrison & Maxwell 1986). As an additional guide to structural disambiguation, the parser includes preference for common word attachment patterns (“tuples”) stored in a small database, constructed manually. Techniques such as (Schubert 2002) suggest ways such a database might be built automatically in future.

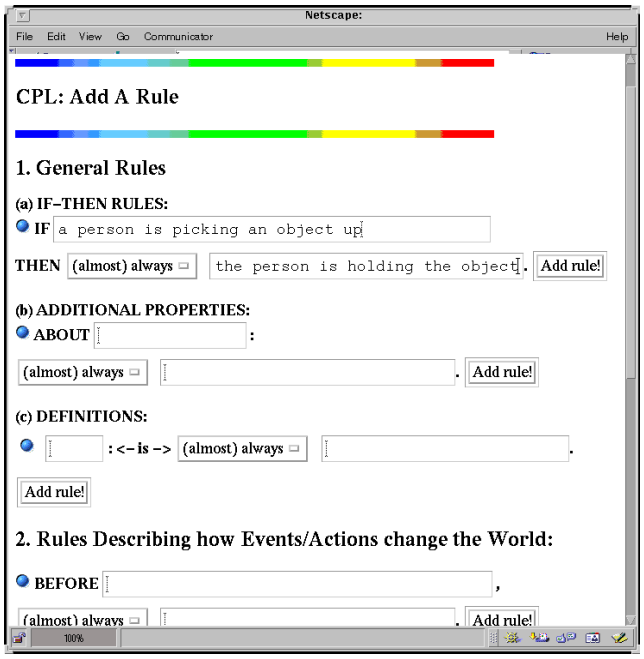

Figure 2: Rules are entered by writing CPL sentences in rule templates.

Logical Form (LF) Generation During parsing, the system also generates a “logical form” (LF). The LF is a simplified and normalized tree structure with logic-type elements, generated by rules parallel to the grammar rules, and contains variables for noun phrases and additional expressions for other sentence constituents. Some disambiguation decisions are performed at this stage (e.g., structural, part of speech), while others are deferred (e.g., word senses, semantic relationships), and there is no explicit quantifier scoping. Some examples of LFs are shown below:

```
;;; LF for “the cat sat on the kitchen mat”
((VAR ?X1 "the" "cat")
 (VAR ?X2 "the" "mat" (NN "kitchen" "mat"))
 (S ?X1 "sit" (PP "on" ?x2))
;;; LF for “an aircraft is transporting objects through the air”
((VAR ?X1 "an" "aircraft")
 (VAR ?X2 NIL (PLUR "object"))
 (VAR ?X3 "the" "air")
 (S ?X1 "transport" ?X2 (PP "through" ?X3)))
```

Generation of KM Sentences Finally, the LF is used to generate ground KM assertions. First, a set of simple, syntactic rewrite rules is applied recursively to the LF to transform it into a set of ground, binary clauses of the form $r(x, y)$[1], where each syntactic relation and preposition becomes a binary predicate. For example, the initial KM for “The cat sat on the kitchen mat” looks as follows:

```
subject(_Sit1, _Cat1)
"on"(_Sit1, _Mat1)
mod(_Mat1, _Kitchen1)
```

[1]In KM this would be expressed as a triple `(x r y)`, but for this paper we will use the standard logical notation $r(x, y)$

Items beginning with a "_" are Skolem constants, each denoting an existentially quantified individual. In this case, there is an instance of a sitting (_Sit1), a cat (_Cat1), a kitchen (_Kitchen1), and a mat (_Mat1), related grammatically as shown. Then, interactively, word senses are assigned to those objects, and syntactic relations are replaced with semantic relations. Finally, other structural reorganizations are performed, including coreference identification and handling coordination.

For word sense disambiguation (WSD), we are currently using WordNet as the target ontology. The word sense disambiguation task is to find a most likely assignment of concepts (i.e., WordNet synsets) to the Skolem constants in the KM, based on the words they correspond to in the original sentence. The system makes a "best guess" of word sense assignments using prior probabilities of single word sense and pairwise word sense assignments for all words/word pairs in the sentence. (The prior probabilities are computed from a tagged corpus). The best assignment is then presented to the user for verification/correction. This approach seems adequate for our purposes, as we allow interactivity; we do not make any strong claims of its accuracy compared with other state-of-the-art WSD systems.

As a special case for WSD, for nominalizations (e.g., "-ing" and "-ion" nouns such as "the falling", "the construction"), the system picks from the verbal rather than noun senses, using the root verb of the nominalization. This allows coreference between verbs and nominalizations to be recognized, and removes the duplication of concepts (verb, verb nominalization) in WordNet from our system.

For semantic relations, the task is to replace syntactic relations in the original KM (verbal subject and object; prepositions; noun-noun relations) with semantically meaningful relations. For our vocabulary of semantic relations, we are using the "slot dictionary" from the University of Texas at Austin[2], a set of about 100 semantic relations. This contains traditional case-role-like noun-verb relations (e.g., agent, patient, instrument), noun-noun modifier relations (e.g., has-part, content, location, material), and event-event relations (e.g., causes, enables). The system picks semantic relations by looking at assignments made in earlier, similar sentences, and hence the system learns relations over time, using a technique similar to (Barker & Szpakowicz 1998).

For example in "travel to a zoo", the initial KM may include the clause `"to"(_Travel01, _Zoo01)`, where `_Travel01` and `_Zoo01` are (after WSD) Skolem instances of the classes (synsets) `travel_v1`[3] and `zoo_v1` respectively, and the task is to replace the syntactic relation `"to"` with the appropriate semantic relation. First the system looks for other, previously processed cases of `"to"` being used between `travel_v1` and `zoo_n1`, and picks the (most common) semantic relation used in those previous cases (here, `destination` would be the desired pick). If none are found, it generalizes the concepts (using WordNet's ontology) and looks for previous cases of `"to"` between `go_v1` and `facility_n1`, etc., iteratively generalizing until some previous cases are found. Again, the user is asked to verify/correct the system's choices. Noun-noun and adjective-noun modifiers are similarly replaced with semantic relations using the same algorithm.

For intrasentence coreference, by default multiple uses of the same word within a sentence are assumed coreferential. To override this, the user includes explicit ordinality markers ("first", "second", etc.), e.g., "A man saw a second man", and in rule (9) in Figure 1.

Finally, structural re-organizations are performed:

1. The verbs "be" and "have" are transformed from event instances to relations, e.g., the clauses `subject(_Be1,_Rose2),object(_Be1,_Red3)` become `"be"(_Rose2,_Red2)`, and subsequently the syntactic relation "be" is replaced with a semantic relation (here `color`) using the earlier technique.
2. Certain other verbs, with senses flagged in the lexicon as "relational verbs", are similarly transformed into a semantic relation, as specified in the lexicon. Examples include (particular senses of) "weighs", "neighbors", and "holds", which transform to the semantic relations `weight`, `neighbor`, and `supports` respectively.
3. Certain nouns, with senses flagged in the lexicon as "relational nouns", are transformed into a semantic relation, as specified in the lexicon. For example, the phrase "the part of a car" generates an initial KM clause `"of"(_Part01, _Car01)`, but will be transformed into the clause `part-of(_Thing01, _Car01)` if the user picks the sense `part_n1` ("something which is a part of something else").
4. One special semantic relation is that of equality, a common meaning for the syntactic relation `"be"`, e.g., in "the color is red". If this relation is chosen, KM will unify (assert as equal) the two equalized objects, e.g., given `equal(_Color1, _Red1)`, this clause will be deleted and all occurrences of `_Color1` be replaced with `_Red1` in the other KM clauses.

After completing the interpretation, the system shows its final interpretation to the user to verify, either displaying the final KM clauses directly or as (rather crudely) paraphrased English, as the user desires. The user can then either accept the interpretation, or modify the original sentence and/or word sense and relation choices and reprocess.

Entering Inference Rules

Basic CPL sentences translate to a set of ground logical assertions between Skolem individuals. To enter universally quantified statements, the user uses one of the seven "rule templates" listed in Figure 3, using the editor shown in Figure 2. The first three of these templates create standard logical implications, and the remaining four describe preconditions and effects of actions, represented using KM's STRIPS-like situation calculus representation of precondition/add/delete lists. As an example of the first template, consider rule (7) from Figure 1:

(7) "IF a person is carrying an entity that is inside a room THEN (almost) always the person is in the room."

[2] www.cs.utexas.edu/users/mfkb/RKF/tree/

[3] For legibility we have renamed the wordnet synset numbers, e.g., 201441983, with more friendly names, e.g., `travel_v1`

Template	**Interpretation**
If *s1* then *typically s2*	$\forall\ x, y\ s1(x, y) \rightarrow \exists\ z\ s2(y, z)$
About *n*: *typically s2*	$\forall\ i\ isa(i, N) \rightarrow \exists\ z\ s2(i, z)$
n/v is *typically np/vp*	$\forall\ i\ isa(i, N) \leftrightarrow \exists\ z\ npvp(i, z)$
Before *s1*, *typically s2*	$\forall\ e, x, y\ s1(e, x, y) \rightarrow precondition(e, \exists\ z\ s2(y, z))$
Before *s1*, it is *typically* not true that *s2*	$\forall\ e, x, y\ s1(e, x, y) \rightarrow precondition(e, \neg\ \exists\ z\ s2(y, z))$
After *s1*, *typically s2*	$\forall\ e, x, y\ s1(e, x, y) \rightarrow add\text{-}list(e, \exists\ z\ s2(y, z))$
After *s1*, it is *typically* not true that *s2*	$\forall\ e, x, y\ s1(e, x, y) \rightarrow delete\text{-}list(e, \exists\ z\ s2(y, z))$

Figure 3: The seven rule templates, and their interpretation in logic. $n/v/np/vp/s$ denotes noun/verb/noun-phase/verb-phrase/sentence respectively. $s1()$ and $s2()$ denote the *set* of clauses produced from sentences $s1$ and $s2$ respectively. Quantification is as in Prolog: All variables in the rule's condition are universally quantified, all remaining variables are existentially quantified. Above, x denotes the *set* of variables which occur only in (the logical clauses produced from) sentence $s1$, y denotes the set of variables which occur in both $s1$ and $s2$, and z the variables in $s2$ only. N is the class (word sense) selected for n/v, and e denotes the main event (the head verb) in (the logical clauses produced from) $s1$. $typically$ is one of "(almost) always", "usually", "sometimes", and "never". Currently it is stored but not used for inferencing.

Before adding quantifiers, the KM clauses which are generated from this (after disambiguation) look:

```
isa(_Person1, person_n1)
isa(_Room2, room_n1)
isa(_Entity3, entity_n1)
isa(_Carry4, carry_v1)
object(_Carry4, _Entity3)
agent(_Carry4, _Person1)
is-inside(_Entity3, _Room2)
 ->
is-inside(_Person1, _Room2)
```

As this is entered as a rule, however, the interpreter modifies its interpretation so that all Skolems in the rule's condition are treated as universally quantified variables (and any remainder in the rule's action as existentially quantified). As in Prolog, this quantification is not explicitly added syntactically, but instead is assumed by the rule interpreter.

Questions

CPL also accepts "wh-" question sentences, such as "What is the man carrying?" and "Who is carrying the red box?". The LF for these sentences contains a "wh-" question variable, denoting the target of the question, e.g.,

```
;;; "Who is carrying the red box?"
((var _x1 "who" nil)
 (var _x2 "the" "box" (an "red" "box"))
 (s (present prog) _x1 "carry" _x2))
```

Questions are handled by transforming them into (the logic for) a definite noun phrase, referring to the answer. After this, the task for the reasoner is to compute the identity of that referent. For example "Who is carrying the red box?" generates (the logic for) "The thing carrying the red box", and "What is the location of the man?" generates "The location of the man". The inference engine then looks up/computes the identity of the referent, using standard reasoning methods of backward chaining and automatic classification. In the above example, the interpreter generates clauses (omitting `isa` clauses):

```
;;; "Who is carrying the red box?"
agent(_Carry2, _Who2)
object(_Carry2, _Box2)
color(_Box2, _Red3)
```

which is then converted to the query:

Find `?Who` where `agent(_Carry2,?Who), object(_Carry2,_Box2), color(_Box2,_Red3)`

Applying the Knowledge Base

In our current system, reasoning with the NL-entered knowledge occurs in the context of a *scenario*. A scenario is a particular situation in the world, denoted by a set of ground assertions in the knowledge base. Scenarios are specified using one or more sentences in CPL, for example, the sentences (1)-(6) shown earlier, and it is against this initial scenario that the inference rules are applied. Rules can be applied in forward-chaining mode, to elaborate an initial scenario with new facts implied but not explicitly stated in the original sentences; or they can be applied in backward-chaining mode, to compute the answer to a specific question posed to the system.

Scene Completion and Semantic Search

One of the applications we are developing is for "semantic search" of a database of video captions. Typically, a caption only explicitly describes a few of the facts about the scene. Many other facts, although "obvious" to a person, are not explicit. For example, consider caption (1):

(1) "A man picks up a large box from a table"

From this caption, a person would also realize that, most likely, the man is holding the box; the box was probably on the table; the man is near the table; the man is lifting the box with his hands; the box is now above the table; the man is standing; etc. In other words, the original caption only gives a partial description of the full scene, and it is our commonsense and domain-specific knowledge of the world that allows us to imagine the "bigger picture". In this application, we are replicating this process by having the inference engine apply rules to the initial (interpreted) caption in a depth-limited, forward-chaining fashion. The result is a much richer caption description, against which search can be performed. For example, a query for "A person holding something" would match the above caption (1), even though there are no words in common, because the enriched caption

description includes the fact that the man is holding the box (inferred from a general rule that picking up something implies (almost always) holding it), and the system also knows a man is a person, and a table is a thing.

Dialog and Simulation

As well as entering a single sentence to describe a scene, our system allows a sequence of sentences to be entered, to describe a "story". As before, each CPL sentence is interpreted interactively. To represent and reason with a sequence of events, CPL uses a standard, STRIPS-like situation calculus mechanism, in which add-lists and delete-lists (entered using the rule templates shown earlier) express the effects of actions on the world, and update the clauses representing the current "situation". During a "story", the initial CPL sentence creates an initial situation. If the next sentence is stative (e.g., "the box is red"), the new facts are added to that situation. If the next sentence is an action though (e.g., "The man moves the box to the floor"), then the STRIPS-like rule for the action (here, "move") is applied to update the situation, reflecting the changes that the action has on the world. In this way, the system tracks the dynamic changes to the world during a story-like dialog. The user can then ask (in CPL) questions of the representation at any time.

Discussion

Our goal is to be able to enter knowledge in a way that is simple, natural, and accessible, and our hypothesis is that a restricted English language, like CPL, reasonably meets this goal. On the positive side, our experience is that CPL is easy to use, and the approach appears to be viable. We currently have a knowledge-base of over 1000 rules and a caption database with over 100 captions, all entered in CPL. The knowledge is inference-capable and easy to inspect and organize, and the system can be easily incrementally developed by gradually expanding the range of language covered.

There are also several challenges presented by this approach. Most significantly, it still takes some skill to use CPL. Simply because the user is authoring in (restricted) English does not mean that he/she no longer needs to care about the underlying processing or semantics. In addition, given that no language interpreter will be 100% perfect, the user needs to learn to "control the beast" to ensure that what the system understands is what he/she intended, and modify the input and/or processing if not. It may take several attempts to enter a rule into the system until it is understood correctly (users familiar with CPL and the interpreter are significantly more efficient).

This control comes in several forms. First, the user needs to reformulate his/her intended rule within the grammatical scope of CPL. In some cases this reformulation task is simple, but in many cases it is non-trivial and requires more than just grammatical conversion. In particular, a "natural" statement of a rule often leaves much knowledge implicit, which needs to be made explicit to produce a meaningful and useful rule in CPL. For example, consider the following glossary definitions taken from WordNet (WN), and an approximate re-expression of them in CPL:

(WN) "attack: intense adverse criticism"

(CPL) "If a person launches an attack against a 2nd person then (almost) always the 1st person criticizes the 2nd person."

(WN) "axis: the center around which something rotates."

(CPL) "If an object is rotating then (almost) always the object is turning around the object's axis."

The point here is that the reformulation task is often more than just rephrasing; it requires making the "natural" language more precise, which in turn requires thinking in a KR-oriented way about the subject matter being described. This itself requires a certain amount of skill and training.

Second, by the nature of natural language, the CPL input is ambiguous in many ways (parse structure, prepositional phase attachment, word senses, semantic relations, bracketing, etc.). The goal of the CPL interpreter is to resolve that ambiguity correctly, but sometimes it makes mistakes, e.g., mis-attaches a prepositional phrase. To handle this, two critical functions need to be supported: First, the user needs to be able to spot the mistake easily, and second, he/she needs to know how to modify the input to correct it. In our current system, the system paraphrases its interpretation back to the user, allowing the user to spot misinterpretations. To correct misinterpretations, the user needs to know a "bag of tricks" for rephrasing the input to avoid specific mistakes. For example, CPL may misinterpret

"the man ate the sandwich on the plate"

as meaning the eating (rather than the sandwich) is on the plate, i.e., misattach the prepositional phrase. A trick to correct this is to rephase using a relative clause:

"the man ate the sandwich that was on the plate"

but the user needs to be aware of such tricks and skilled in their use. A challenge for languages like CPL is to devise methods so that these corrective strategies are taught to the user at just the right time, e.g., through the use of good system feedback and problem-specific on-line help.

A third issue, closely related to the reformulation task mentioned earlier, is that CPL has limited expressivity compared with the (rich) expressive power of the underlying KR language, KM. While one might think that, intuitively, natural-language-based knowledge is more expressive than traditional KR, we believe the opposite is true: one must distinguish between human expressivity and computational expressivity, and it is computational expressivity – the amount the computer understands, i.e., is able to use to draw sensible conclusions from – that matters. With no computational ability to manipulate "represented" knowledge, the representation really has no more status than a comment (otherwise, one could argue that the Web has already "solved" KR). From our perspective, there are numerous representational phenomena that can be directly expressed and reasoned with in the underlying logical formalism KM, but which are currently outside the scope of CPL and its interpreter. These include constraints, defaults, other quantification patterns, and mathematical expressions.

A fourth, interesting issue concerns the adequacy of linguistically motivated representations. In a non-linguistically-motivated knowledge base (KB), one some-

times finds concepts and theories without any obvious, direct linguistic counterpart, introduced to achieve particular inferential goals, but these will be largely absent from a language-generated KB. Do we pay a price for this absence? For example, a CPL-generated representation of "walked for 10 miles" would look:

```
distance(_Walk1,_Mile1)
count(_Mile1,10)
```

whereas conventional KR wisdom would suggest distinguishing the physical quantity from its magnitude on a measuring scale, e.g., representing this as:

```
distance(_Walk1, _Distance1)
value(_Distance1, 10, mile)
```

The latter representation makes a subtle but critical distinction, enabling the computer to easily compare distances measured in different units, describe the same distance using multiple, different units, perform unit conversion, etc. In the CPL representation, however, these tasks would be more difficult to perform because the representation confuses this distinction. Because the grammatical structure of the language does not point to this distinction, it is lost. Of course, the CPL author could write a somewhat artificial-sounding CPL sentence to make that distinction, but then the author would need to be a knowledge engineer to realize it needed to be made in the first place, which would be self-defeating.

Finally, and closely related, is the issue of *canonicalization*. In English, typically there are several ways of saying the (approximately) same thing (e.g., "conducting a test of an entity" and "testing an entity"), while in a hand-designed ontology typically there is one prescribed way of expressing that knowledge. If a natural-language-based KR system naively translates each English variant to a different symbolic variant, then the proliferation of variants will persist, and without knowledge of their equivalence the computer may miss conclusions that it should otherwise draw. In principle, one could add rules to explicitly state these equivalences (e.g.: "IF an agent is conducting a test of an entity THEN the agent is testing the entity."), but in practice this is probably infeasible due to the large number of such rules required. Rather, what is required is some normalization of the input, by the interpreter itself (e.g., automatically converting "conducting an X of Y" to "Xing Y"), by restricting the vocabulary and grammatical forms allowed (e.g., don't allow the verb "conduct"), and by harnessing expectations from the knowledge base to guide interpretation. In our existing work with CPL we have directly experienced this problem (CPL is perhaps too permissive at the moment), and we are pursuing all these ways to deal with it. In particular, we consider harnessing prior ontological expectations as particularly important for addressing many of the former issues. Given a pre-defined a ontology, the system can check its CPL interpretation against it and if it does not "make sense", can look at modified interpretations. In this way, a pre-defined ontology can channel the interpretation into a canonical and well-principled form, a task closely related to handling metonymy (Fass 1991; Fan & Porter 2004), and one we consider critical for future evolution of this technology.

Conclusion

We have presented CPL, a restricted English language for expressing world knowledge, and have described how CPL is interpreted and used for reasoning. Restricted English is an exciting way forward, both as a means for widening the "knowledge acquisition bottleneck" and making inroads into the long-term goal of understanding unrestricted text. Most importantly, we believe it breaks a passage through the seemingly impassable extremes of hand-coding directly in logic, and understanding unrestricted English.

Despite this, as we have discussed, there is no "free lunch" by using a natural-language-based approach to KR. Issues of semantics and representation need to be addressed either by the knowledge author him/herself or by the machinery interpreting the language input. Nevertheless, we have found that with some training and experience, the approach is viable. We have successfully constructed an extensive, inference-capable knowledge base of over 1000 rules using this technology, which is being used in an experimental application for semantic search of video captions. Given the growing demand for semantically meaningful annotations from the Semantic Web and information retrieval community, and the long-standing need for more accessible knowledge acquisition technologies in AI, we believe that processing restricted language has significant future potential for the field.

References

Barker, K., and Szpakowicz, S. 1998. Semi-automatic recognition of noun modifier relationships. In *Proc. COLING-ACL'98*, 96–102.

Clark, P., and Porter, B. 1999. KM – the knowledge machine: Users manual. Tech report, Univ Texas at Austin.

Fan, J., and Porter, B. 2004. Interpreting loosely encoded questions. In *AAAI'04*.

Fass, D. 1991. met*: A method for discriminating metonymy and metaphor by computer. *Computational Linguistics* 1(17):49–90.

Fuchs, N. E.; Schwertel, U.; and Schwitter, R. 1998. Attempto controlled english - not just another logic specification language. In Proc. LOPSTR'98.

Harrison, P., and Maxwell, M. 1986. A new implementation of GPSG. In *Proc. 6th Canadian Conf on AI*, 78–83.

Hoard, J.; Wojcik, R.; and Holzhauser, K. 1992. An automated grammar and style checker for writers of Simplified English. In Holt, P., and Williams, N., eds., *Computers and Writing: State of the Art*, 278–296. UK: Intellect.

Mitamura, T.; Baker, K.; Nyberg, E.; and Svoboda, D. 2003. Diagnostics for interactive controlled language checking. In *Proc. Controlled Language Application Workshop (CLAW'03)*.

Schubert, L. 2002. Can we derive general world knowledge from texts? In *Proc. HLT-02*, 84–87.

Schwitter, R., and Tilbrook, M. 2004. PENG: Processable ENGlish. Technical report, Macquarie Univ, Australia. http://www.ics.mq.edu.au/∼rolfs/peng/.