Naturalness vs. Predictability: A Key Debate in Controlled Languages

Peter Clark, William R. Murray, Phil Harrison, John Thompson

Boeing Research and Technology, PO Box 3707, Seattle, WA 98124, USA
{peter.e.clark, william.r.murray, philip.harrison,john.a.thompson}@boeing.com

Abstract. In this paper we describe two quite different philosophies used in developing controlled languages (CLs): A "naturalist" approach, in which CL interpretation is treated as a simpler form of full natural language processing; and a "formalist" approach, in which the CL interpretation is “deterministic” (context insensitive) and the CL is viewed more as an English-like formal specification language. Despite the philosophical and practical differences, we suggest that a synthesis can be made in which a deterministic core is embedded in a naturalist CL, and illustrate this with our own controlled language CPL.

In the second part of this paper we present a fictitious debate between an ardent “naturalist” and an ardent “formalist”, each arguing their respective positions, to illustrate the benefits and tradeoffs of these different philosophies in an accessible way.

Part I: The Naturalist vs. Formalist Debate

1 Introduction

There are two quite divergent schools of thought concerning the design of controlled languages. The first, which we call the **"naturalist"** approach, treats CL interpretation as a simpler form of the full natural language (NL) processing task in which ambiguity still resides, only to a lesser extent. One might say that the goal is to make English more tractable (understandable by computers) by simplifying the complexity of the English handled. In this approach, as with full NL, multiple interpretations of a sentence are possible, and the CL interpreter uses all the standard NL machinery to search for a "best" parse and interpretation, but with the task being constrained to just the subset of English covered by the CL. Examples of “naturalist” CLs include our own, called CPL [1] (described later), and (to a degree) CELT [2].

The second approach, which we call the **"formalist"** approach, views a CL more as an English-like formal/programming language that is well-defined, predictable, and easier to use than a normal formal language. One might say that the goal is to make logic more tractable (easier to use by humans) by rendering it in human-readable terms that non-mathematicians can understand. Given that many existing formal languages are somewhat cryptic to untrained users, and good NL processing techniques already exist, it is a natural step to have users work with CLs instead that are more readable and translate deterministically into the formal language. A "formalist" approach would view this as an end in itself, and make no claims that the CL necessarily says anything about NL processing in general. Examples of “formalist” CLs include ACE [3], PENG [4], CLCE [5], and CLIE [6].

Although it might seem that the "naturalist" and "formalist" approaches are just variants for tackling the same overall problem, their underlying philosophies are quite different. A formalist approach eschews nondeterminism, requiring that the CL translates cleanly and predictably to a formal representation, i.e., there is only one acceptable parse and interpretation for any sentence, there is a single sense for each word (plus part of speech), and interpretation follows a logically defined path. In contrast, a naturalist approach attempts to do NL interpretation "in the small," making many disambiguation decisions heuristically (e.g., prepositional phrase attachment, semantic role labeling), and searching for an overall "best" interpretation. A naturalist might seek to gradually extend the CL with time, taking small steps towards fuller NL understanding, while a formalist might declare the CL complete once sufficient expressivity had been achieved.

These different approaches can produce quite different results. A naturalist CL may be more fluent/natural for the user, but also harder to control because the user cannot always predict the disambiguation decisions that the system will make. Conversely, a formalist CL may be easier to control (once the user has learned the disambiguation rules), but may also be somewhat less natural to read and may require the user to understand more about the target representation language and ontology.

At Boeing we have been developing a controlled language called CPL (Computer-Processable Language) which clearly falls in the "naturalist" category. We have also created CPL-Lite, a more constrained version which (unlike CPL) is deterministic and falls into the "formalist" category. In this paper, we describe these two languages and discuss the strengths and weaknesses of each. We then suggest a way in which the dichotomy between these two rather different approaches might be resolved, and describe ongoing work at creating such a combination.

Following this, the second half of the paper presents a debate between a “naturalist” and a “formalist”, each arguing his position. Although the dialog is fictitious, it draws on material from numerous discussions over the years, including at the CNL 2009 Workshop. The goal of presenting the debate is not to argue one position over the other, but rather to highlight the strengths and weaknesses of both philosophies in an easily accessible way.

2. Two Controlled Language Variants

2.1 CPL - Computer-Processable Language

CPL is a mature and extensive controlled language, used in several projects [1,7]. It is clearly in the “naturalist” category as it attempts to do simplified natural language processing, using a variety of heuristics to make disambiguation decisions and produce the “natural” or “obvious” interpretation that a person would expect.

Briefly, CPL accepts three types of sentences: ground facts, questions, and rules. For ground facts, a basic CPL sentence takes one of three forms:

> "**There is|are** *NP*"
> *"NP verb [NP] [PP]*"*
> *"NP* **is|are** *passive-verb [***by** *NP] [PP]*"*

where *verb* can include auxiliaries and particles, and nouns in *NP*s can be modified by other nouns, prepositional phrases, and adjectives. For questions, CPL accepts five forms, the two main ones being:

> "**What is** *NP***?**"
> "**Is it true that** *Sentence***?**"

For rules, CPL accepts the sentence pattern:

> "**IF** *Sentence [***AND** *Sentence]** **THEN** *Sentence [***AND** *Sentence]**"

CPL's grammar supports simple sentences, prepositional phrases, compound nouns, ordinal modifiers, proper nouns, adjectives, pronouns, definite reference, and a limited use of conjunction (allowing fillers of the same semantic role to be conjoined, e.g., "John met Sue and Mary"). It does not allow adverbs, modals, relative clauses, imperatives, disjunction, or other forms of coordination. For (non-unitless) physical quantities, e.g., "10 meters", a unit of measure must be given.

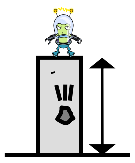
Definite reference is resolved by searching for the most recent, previously mentioned object with the same description, e.g., "The red block..." will resolve to the most recently mentioned red block. Ordinal reference, e.g., "The second red block...", behaves similarly except counts referents starting from the beginning of the CPL text. If a referent is not found, CPL assumes the reference is to an implicitly existing object and so creates that object in its internal interpretation. Pronouns resolve to the most recent object with the same gender as the pronoun.

Existential quantification is expressed using CPL sentences, and these always translate to Skolemized (i.e., existential) ground assertions. Universal quantifiers such as "every", "all", "some", and "most" are not allowed. Universal quantification is expressed using CPL "IF…THEN…" rules, and quantifiers are applied in the same way as in Prolog (namely variables in the "action" part of the rule are universally quantified, and remaining variables are existentially quantified).

CPL is designed to work with a pre-defined ontology (including words associated with each concept and relation in the ontology), the primary one being UT Austin's Component Library ontology [8], although others can be used too. Words outside the target ontology's associated lexicon can be used, in which case CPL uses WordNet and the target ontology's lexicon to find the closest concept to those words. CPL does not currently support extending the lexicon or ontology using CPL statements.

The most extensive application of CPL to date has been for posing exam-style questions to AURA [7], a knowledge-based system containing formal knowledge about physics, chemistry, and biology [9,10]. To pose questions, users reformulate the original English in CPL, aided by a set of guidelines, examples, and on-line feedback when invalid CPL is entered. For example, given the original AP exam question:

> An alien measures the height of a cliff by dropping a boulder from rest and measuring the time it takes to hit the ground below. The boulder fell for 23 seconds on a planet with an acceleration of gravity of 7.9 m/s^2. Assuming constant acceleration and ignoring air resistance, how high was the cliff?



A typical way that the user might express this in CPL would be:

> An alien drops a boulder.
> The initial speed of the boulder is 0 m/s.
> The boulder drops for 23 seconds.
> The acceleration of the boulder is 7.9 m/s^2.
> The distance of the drop equals the height of the cliff.
> What is the height of the cliff?

As can be seen, the translation from the original English to CPL requires some work by the user, both conceptually (to factor out some of complex notions in the original English that are outside the expressive capability of AURA, and to make commonsense knowledge explicit), and linguistically (to simplify the sentence structure). However, given a good set of worked examples, on-line feedback by CPL, and a small (4 hours) amount of training, users were able to master this task during our evaluations [7], although typically making several attempts before finding a valid and adequate CPL formulation.

CPL is implemented as the vocabulary and grammar restrictions described above placed on the underlying language interpreter BLUE (Boeing Language Understanding Engine). Sentence interpretation is performed using a 10-step pipeline:

1. **Preprocessing** – remove non-ASCII characters etc.
2. **Parsing** – using SAPIR, a broad-coverage chart parser [11]
3. **Syntactic logic generation** – convert parse tree to “syntactic” logic where predicates and arguments are words or Skolems, not concepts.
4. **Reference resolution**
5. **Transform Verbs to Relations** – convert some reified verbs to predicates
6. **Word Sense Disambiguation** (WSD) – using WordNet to help pick senses, thus replacing word arguments with concepts from the target ontology.
7. **Semantic Role Labelling** (SRL) – using a hand-crafted set of ~100 rules to replace word predicates with semantic relations.
8. **Metonymy resolution** – using 5 metonymy rules e.g., PLACE for OBJECT
9. **Question Annotation** – tag the query variables
10. **Additional Processing** – miscellaneous tidying up

Further details are given in [12]. Heuristic rules are used for prepositional phrase attachment, word sense disambiguation, semantic role labeling, compound noun interpretation, metonymy resolution, and other language processing activities. Output is in a Prolog-like syntax. A simple example of the input and output is as follows:

```
;;; CPL: "A person throws a ball from the top of a building."
 isa(person01,Person),
 isa(ball01,Hollow-Ball),
 isa(top01,Region),
 isa(building01,Building),
 isa(throw01,Throw),
 has-region(building01,top01),
 agent(throw01,person01),
 object(throw01,ball01),
 origin(throw01,top01).
```

2.2 CPL-Lite

CPL-Lite was designed for a separate project called Mobius [13] to allow trained knowledge engineers, who understood the target knowledge base, to pose queries in a way that was controllable and also (reasonably) comprehensible to others. While CPL searches for the best interpretation, CPL-Lite is simpler and interpreted deterministically (no search or use of heuristics).

CPL-Lite uses the same interpreter as CPL, but with additional restrictions and modifications. First, the grammar is substantially further restricted to just a set of template sentences (described shortly), thus avoiding ambiguities that would otherwise arise. Second, the vocabulary is restricted to just words that map directly to a concept in the ontology (WordNet is not used to "guess" the meaning of other words). Third, CPL's heuristics are switched off and just a default behavior is used: word sense disambiguation (WSD) simply picks the a priori most likely word sense; semantic role labeling (SRL) rules are fixed and defined by the template sentences (below); and metonymy resolution rules are not used. Although these restrictions may seem severe, bear in mind that CPL-Lite is designed to be used by a knowledge engineer with full knowledge of the ontology its vocabulary, rather than an end-user.

For the sentence templates, each CPL-Lite sentence corresponds to a single binary relation (i.e., slot, predicate) between two entities. For assertions, there are 113 sentence templates of three types (below), depending whether the relation appears in language as a noun, verb, or preposition:

CPL-Lite Sentence Pattern	**Interpretation**
For the 82 noun-like relations:	
“The age of a *entity* is a *duration*.”	→ age(*entity’*,*duration’*).
“The agent of a *event* is a *entity*.”	→ agent(*event’*,*entity’).*
…etc…	
For the 10 verb-like relations:	
“A $event_1$ causes a $event_2$.”	→ causes($event_1$’,$event_2$’).
“A $entity_1$ encloses a $entity_2$.”	→ encloses($entity_1$’, $entity_2$’).
…etc…	
For the 21 preposition-like relations:	
“A $entity_1$ is above a $entity_2$.”	→ is-above($entity_1$’, $entity_2$’).
“A $entity_1$ is behind a $entity_2$.”	→ is-behind($entity_1$’, $entity_2$’).
...etc..	

where e*ntity’* is the interpretation of (i.e., instance denoted by) “a *entity*”, etc.

The significance of this is that it allows any of the 113 predicates in the underlying ontology to be expressed unambiguously, and thus allows the user to “force” a particular interpretation in a formalist-style manner (although the CPL-Lite English will thus be more verbose than the corresponding CPL, as only one relation is expressed per sentence). *NP* is one of ~1000 simple nouns (including a few compound nouns) that map directly to concepts in the target ontology, i.e., is in the ontology's associated lexicon. Complex noun phrases are not allowed. In addition, the sentence form "*NP verb [NP]*" is allowed, where *verb* is in the ontology's lexicon (mapping to an action/event concept), and with the interpretation that the first and second NP are always the agent and object of the verbal concept respectively, i.e., are interpreted as agent(x,y) and object(x,y). Definite reference, “IF…THEN…” rule sentences (containing CPL-Lite sentences only), and three question forms are also supported in CPL-Lite.

3. Comparison, and the Naturalist vs. Formalist Tradeoff

Interestingly, CPL-Lite has the same expressivity as CPL; it is just more verbose and grammatically restricted, and requires more knowledge of the vocabulary and structure of the target ontology. It is also more predictable: A knowledgeable user can enter a sentence and know exactly how it will be interpreted. In contrast, the “naturalist” language CPL is more fluent and tolerant of the user: it uses WordNet to "guess" meanings for unknown words, will use lexical and semantic knowledge to try and perform PP attachment and semantic role labeling correctly, and attempt to resolve metonymy. However, as the CPL interpreter is more complex than that for the “formalist” language CPL-Lite, there is arguably more of a risk that CPL may not interpret the sentence in the way the user intended, and it may not be obvious to him/her how to reformulate the CPL to correct the error (if indeed the user is aware of the misinterpretation). To handle this, CPL paraphrases and graphs its interpretation back to the user so he/she can validate/correct the system’s understanding.

To illustrate the naturalist/formalist distinction, consider an example of CPL and the corresponding CPL-Lite, taken from the AURA application [9]. In this example, the user describes part of a physics problem in CPL as follows:

> A man drives a car along a road for 1 hour.
> The speed of the car is 30 km/h.

When processing this, CPL interprets "for" here to mean the predicate duration(), "along" to mean path(), and attaches both prepositional phrases in the first sentence to the verb ("drive"). In addition, the target ontology considers speeds as properties of events, not objects, and so CPL interprets the second sentence as metonymy for "The speed of the car’s *driving* is 30 km/h" (i.e., it resolves the metonymy with respect to the target ontology). Finally, the concepts of “car” and “man” are not part of the CLib ontology, so CPL climbs the WordNet taxonomy to find generalizations which are in the ontology – here Vehicle and Person – and maps these words to those concepts, i.e., tries to find the nearest concept which is known. The resulting interpretation looks:

```
isa(drive01, Drive),
isa(man01, Person),
isa(car01, Vehicle),
isa(road01, Road),
agent(drive01, man01),
object(drive01, car01),
path(drive01, road01),
duration(drive01, [1,hour]),
speed(drive01, [30,km-per-hour]).
```

Note that the same resulting interpretation[1] could have been produced by a CPL-Lite formulation as follows:

> A person drives a vehicle.
> The path of the driving is a road.
> The duration of the driving is 1 hour.
> The speed of the driving is 30 km/h.

In this CPL-Lite formulation, the user had to explicitly identify and spell out the target concepts (e.g., Person); had to explicitly spelled out (the words corresponding to) the target predicates; has removed the PP attachment ambiguity; and has correctly (with respect to the constraints in the target ontology) attached the speed to the driving event (rather than the car) as required by the ontology.

The important point to note is that both formulations produce the same resulting logic. To write in the formalist CPL-Lite style, the user needs a clearer picture of the target representation he wishes to build, i.e., needs to know the specific concepts and predicates he wishes to use, how they can be combined, and what words can be used to refer to them. He also needs to write in a more verbose and, here, somewhat less natural way. However, the authoring task is also quite straightforward – the user writes in simple, unambiguous English and can easily predict how it will be interpreted. In contrast, the “naturalist” CPL has the advantage of perhaps being more fluent, but also carries an increased risk of being misinterpreted. For example, consider the (hypothetical) case that the CPL interpreter misinterprets "for" in the earlier CPL sentence:

> A man drives a car along a road for 1 hour.

as meaning beneficiary(x,y) (i.e., the hour is the “beneficiary” of the driving). In this case, CPL's "smarts" have gone wrong, and the user is left either unaware of the error, or aware of it but unsure how to rephrase the sentence to correct the problem. To mitigate this problem, CPL paraphrases back its understanding in CPL-Lite and also as a graph so the user is aware of that understanding, and provides reformulation advice and a library of CPL examples to help the user reword if necessary. In general, though, "smart" software is a mixed blessing -- it can be very helpful, or frustrating to control, or both (e.g., consider automatic page layout or figure placement software).

[1] bar different arbitrary Skolem names. Note man01 is an instance of Person, not man, as there is no concept of a man in the target ontology.

This is essentially the tradeoff that the naturalist vs. formalist approaches presents. We illustrate these tradeoffs further in the dialog in part two of this paper.

4. Synthesis: Combining the Two Approaches

While it may seem that the underlying philosophies of the naturalist vs. formalist approaches are incompatible, there is a synthesis which we have made, namely to "embed" CPL-Lite as a deterministic core within CPL itself. In other words, within CPL, we have included the core set of the 113 CPL-Lite sentence patterns described earlier whose interpretation is deterministic (i.e., context-insensitive) and easily predictable. As CPL-Lite already uses the same interpreter as CPL, the mechanism for this "embedding" is straightforward, namely to ensure that when the CPL-Lite sentence templates are used, only CPL's default behavior rather than additional heuristics are applied. Thus only a small change to the existing CPL software was needed to ensure that no "fancy heuristics" were triggered by use of the CPL-Lite templates.

Given such a core, the user can then work within it or beyond it to the extent that he/she feels comfortable, and can fall back on the core if the "smart" interpretation goes wrong. For example, should the CPL interpreter have misinterpreted "for" in "drives...for 1 hour" as beneficiary(x,y), the user can revert to CPL-Lite to make the desired relation explicit: "The duration of the driving is 1 hour", thus correcting the mistake. In this way, the user both has the flexibility to write in more fluent English, while also being able to resort to a more constrained form when necessary to control the interpretation.

Over the last two years we have used CPL, including the deterministic CPL-Lite core, in the AURA application described earlier, including evaluating its performance [14]. One striking feature of CPL's usage in this context is that although CPL provides for a wider variety of forms than CPL-Lite, users typically stay within the CPL-Lite subset in the majority (~70%) of their sentences. The most common examples of going beyond CPL-Lite were using complex noun phrases (e.g., "the direction of the applied force on the object"), using words outside the KB's lexicon (e.g., "car", "horse"), and using metonymy with respect to the KB (e.g., "The speed of the man" for "The speed of the man's movement"). As the users were trained with largely CPL-Lite-style example sentences it is perhaps not surprising that they often stayed within this subset, and did not often venture into more sophisticated language forms, and when they did venture out it was somewhat conservatively. This suggests that for users to feel comfortable going beyond that core, the quality of the interpretation needs to be high, and thus "naturalist" CLs are perhaps mainly appropriate for domain-specific applications where the required level of domain-specific customization can be made.

5. Summary

Although the philosophies underlying the naturalist and formalist approaches differ, there is a strong case to be made that each needs and can benefit from the methods of the other. For a naturalist CL such as CPL, there still needs to be a way for the user to control the interpreter's behavior when he/she knows what target output is needed, and this can be done by embedding a formalist-style core, as we have done by embedding CPL-Lite as a core of CPL. Conversely, for a formalist CL such as CPL-Lite, the CL can sometimes be verbose and disfluent, and require a deeper understanding of the target ontology. Adding a more naturalist-like layer to the CL can alleviate these problems, providing that there is sufficient feedback to the user so that there is no confusion about what the system understood. In fact, in practice there are degrees of non-determinism that might be introduced or rejected (e.g., grammar; word senses; semantic roles), and so in implementational terms there is perhaps more of a continuum between the naturalist and formalist extremes. Thus although there are two quite different philosophies underlying the design of modern CLs, each has substantial benefits to be gained from the other, and there are good reasons for expanding the dialog between practitioners of each.

Part II: A Debate between a Formalist and a Naturalist

As we have discussed, the tradeoffs between the formalist and naturalist positions are quite complex, and there are strong arguments both for and against each position. To make these more vivid, and also capture some of the discussions both before and during the CNL 2009 Workshop, we conclude with a fictitious debate between a "formalist" (**Form**) and "naturalist" (**Nat**) about their respective positions. It integrates several real discussions and exchanges over the years, including from the CNL'2009 Workshop, on the topic of the naturalness vs. predictability tradeoff. The discussion is not meant to reach a conclusion, but rather highlight the points made by proponents of both philosophies in an accessible and balanced way.

[Predictability vs. Naturalness]

Form: Well, as a firm believer in the "formalist" approach I'm not sure I buy all of this. It seems essential to me that CNLs should be *unambiguous*, i.e., for any sentence, there is *just one and only one valid interpretation*. CNLs are, by definition, intended to be a precise means of communication between the user and the machine, and if we allow ambiguity in the process then we will have reintroduced precisely the problem that CNLs are meant to avoid. Parsing, for example, should be deterministic - there should be only one valid parse for any given sentence. In that way, the user has complete control over how his input is interpreted, and can create the target formal representation he intends.

If we allow non-determinism, ambiguity, and heuristics in the CNL interpreter, how is the user ever going to ensure that what he says is interpreted in the way he intends? If we try and make the software too clever, it will only end up confusing the user. Predictability is better than smarts!

Nat: I think the claim of "no ambiguity" is misleading: Ambiguity is a property of language, not of a language interpreter, and it is just a fact that most English statements are inherently ambiguous. Any CNL, formalist or naturalist, has to resolve that ambiguity in some way. If a "formalist" CNL only "sees" a single interpretation, then it still has made a choice, implicitly in its design, not to consider any alternative interpretations. But to the average user those alternatives are still there.

Form: Well, maybe the phrase "no ambiguity" isn't the best, but let me try to be clear what I mean: I believe a CNL should have a "deterministic" grammar, i.e., a small set of grammar rules that do not conflict with each other or offer multiple alternatives. The grammar should only ever admit *one* interpretation, so it is completely clear to the user how a sentence will be interpreted.

For example, we might have a rule that prepositional phrases always attach to the preceding verb (an application of the minimal attachment principle). Thus, given a potentially ambiguous sentence, it is now completely clear how the CNL will understand it.

Nat: Completely clear to who? Suppose the user writes:

(1) The man lives in the house with a red roof.

It is "completely clear" to the user that the house has a red roof (rather than that the man lives with a red roof). But a CNL using your attachment rule will not interpret the sentence this way.

Form: Well, the user has to learn the grammar rules, and consider how they will be applied as he authors CNL sentences.

Nat: That's a lot of work for the user!

Form: So what? Users are smart and can be trained!

Nat: But why make things so hard for the user? In the end, all the user wants is to be understood correctly. If he writes:

(1) The man lives in the house with a red roof.

he wants the computer to understand the house has a red roof, while if he writes

(2) The man lives in the house with his wife.

then he wants the computer to understand that the man is living with his wife.

Form: Well, if he wants to say (1) then he should simply phrase the knowledge in a different way, e.g.,

(3) The man lives in the house that has a red roof.

That's not a big deal.

Nat: But how does the user know that (1) needs to be rephrased?

Form: He learns the CNL rules, or we can have the CNL paraphrase its interpretation back to the user[2] so he can see whether it was understood in the way he intended.

Nat: This seems to be making life very hard for the user. Look, we want to enable the user to write as naturally as possible. While this is too difficult to implement for full natural language, the beauty of CNLs is we can restrict the language to a point where natural authorship and interpretation is possible. Why make the

[2] e.g., see Kaljurand's paper [15] elsewhere in these proceedings.

user do this kind of rephrasing (3) you suggest if we can build CNL interpreters that can understand the original (1) in the first place?

Form: No, I disagree. We want the users to write as precisely as possible, and encourage them to be clear and unambiguous. If the system is full of heuristics, how is the user going to know how any CNL statement he writes will be understood? Suppose the heuristics go wrong? Suppose a statement is misinterpreted?

Nat: Well, as you yourself just suggested, the CNL interpreter can paraphrase back its understanding to the user so he can validate that the computer has understood correctly. We could also use tools to help the user construct only sentences that are valid[3]. And this requirement for user validation isn't just for a "naturalist" CNL - what happens if a "formalist" CNL doesn't interpret the user's sentence the way he expects, e.g., because he hasn't fully internalized the rules, or the CNL's lexicon is missing something? *Any* usable CNL is going to need some way to allow the user to see and validate/correct the system's interpretation.

Form: I think the user is going to make fewer mistakes using a "formalist" CNL, because he can predict how his sentences will be interpreted.

Nat: No, I would say the opposite is true - I think the user is going to make fewer mistakes using a "naturalist" CNL, because the system's interpretation rules are closer to those of natural language, and so the system is more likely to do what the user naturally expects.

Form: Hmmm….these conjectures would be interesting to test.

Nat: Indeed! Perhaps your approach might best suit users trained in logic where their ultimate task is to write specifications. And mine might best suit lay users who just want to *use* a system by interacting with it in English.

[Lexical Ambiguity and Word Sense Disambiguation]

Form: Here is another important aspect of many "formalist" CNLs: each word (in a given part of speech) should have only one sense - the "one sense per word" principle. This I believe is important, because then the user does not have to worry about word sense disambiguation. Rather, he just needs to use words consistently. The words he uses becomes the ontology of the theory he builds.

Nat: That's a big restriction! That might work fine for nouns and verbs, if the user is careful, but what about for prepositions? For example, given

(4) The roof of the house...

what predicate would "of" translate to?

Form: It just becomes the predicate: of(x,y).

Nat: But what does of(x,y) *mean*?

[3] e.g., menu-driven sentence builders such as NL-Menu [16], the Ace Editor and AceWiki [17], and ECOLE used in PENG [18].

Form: It means whatever the user says it means. The user would write whatever rules (in CNL) he felt was appropriate to capture the meaning. Again, we want the user to have complete control, rather than be shoe-horned into a pre-defined ontology.

Nat: But "of" can mean many things, e.g.,:

(4) The roof of the house... [part of]
(5) The cellphone of the girl... [possessed by]

"of" clearly corresponds to a different semantic relation in (4) and (5); you can't just translate all of these to: of(x,y)! Suppose I want to write axioms about the "part of" relation, for example? Those axioms should apply to (4) but not (5), but if both are translated to of(x,y) then the important distinction is lost.

Form: Well, my first reaction is to say the user shouldn't write such ambiguous sentences in the first place. Instead, he should make the relation explicit, for example instead of (4) and (5) he should write:

(6) The roof that is a part of the house...
(7) The cellphone that is possessed by the girl...

Alternatively, the user can write axioms for of(x,y) directly by including extra conditions in those axioms. For example, if the user wants to write axioms for cases where of(x,y) means is-part-of(x,y), he can add conditions to select just those cases where "of" means "part of". He might decide that if x and y are artifacts, then "x of y" means "x part-of y", and so can write part-of axioms using "of" plus an "artifact" test:

(8) **IF** there is an artifact X **of** an artifact Y
THEN <*some assertion about part-whole relationships*>.

Nat: Ugh! Your first alternative, (6) and (7), requires the user to mentally do preposition disambiguation himself. How will he even notice when it needs to be done? The second alternative, (8), looks even worse - the user is essentially having to write semantic role disambiguation rules within all his axioms!

Form: Sure, why not? The user *should* be doing the disambiguation himself and *should* be precise in his CNL sentences. How else is he going to reliably communicate with the machine?

Nat: That's placing a huge burden on the user, making him write in a less natural and more convoluted way.

Form: But that's a good thing, if that natural way is unclear or ambiguous.

Nat: No, that's a bad thing - the user would be happier if he could write naturally and have the computer work out the meaning itself! For example, he'd like to write:

(4) The roof of the house...

and have the CNL interpret this as, say, has-part(house01,roof01).

Form: I think that takes too much control away from the user, and unnecessarily forces him into a particular target ontology.

Nat: Or maybe that's helping the user, as the CNL interpreter is doing some of the work for him.

Form: Or maybe that's not helping the user, if the CNL interpreter's heuristics go wrong. The user will be happiest if the system is easy to control, and if he can easily predict what it will do with his sentences.

[Canonicalization]

Nat: A similar issue occurs with common, highly ambiguous nouns and verbs. Consider the meaning of the verb "be" in the following sentences:

(9) The box is red.
(10) The color of the box is red.

In (9), "is" is best interpreted as something like has-color(x,y), while in (10), "is" denotes equality, =(x,y). Ideally, a CNL interpreter would make such determinations automatically. If it does this, then the interpretations of (9) and (10) will be the same although the surface syntactic forms differ. This is desirable, as (9) and (10) are essentially different ways of saying the same thing.

Form: But who says that has-color(x,y) is the right interpretation of "is" in (9)? If the CNL interpreter makes this interpretation, then it forced an ontological commitment - that the ontology should include has-color(x,y) - on the user. This seems undesirable to me. Again, the user, not the system, should decide on the ontology to use and be in control. In addition, (9) and (10) together violate the "one sense per word" principle, and thus is a potential recipe for confusion for the user.

Nat: But is that really a problem? It seems more of a problem to me to have lots of be(x,y) predicates in the interpretation, all meaning different things.

Form: It is the user's job to be consistent about how they use a predicate like be(x,y), and the CNL's job to be consistent in the interpretation.

Nat: That seems pretty constraining to me, when we're talking about ambiguous verbs like "be" or "have".

Form: Users are smart, they can work it out and be precise!

Nat: But why impose such constraints in the first place? There has been substantial progress in language processing technology over years that can be exploited, so that users can write more naturally.

Let's discuss another example of canonicalization, namely the interchangable use of verbs and their nominalizations[4]. Consider:

(11) The army destroyed the city at night.
(12) The destruction of the city by the army was at night.

[4] See [19] for a good discussion of this topic.

Again, as (11) and (12) mean essentially the same things. It would thus be nice if the CNL could produce the same canonical representation.

Form: Ugh, that is making the interpreter way too complex and unpredictable. Basically, the more fancy stuff the interpreter does, the less the user knows what to expect. Better to have a more surface-level, direct mapping from the sentences to the logic that the user can get his head around.

Nat: What about actives and passives?

(13) A man loves a woman.
(14) A woman is loved by a man.

Should these produce different interpretations?

Form: Well, sure, those can be mapped to the same representation.

Nat: So some canonicalization is ok?

Form: Sure, if it's simple cases like that.

Nat: Where do you draw the line?

Form: Well, some minimal canonicalization like that is okay; the important thing is that it is pretty minimal and easy for the user to grasp.

[Use of a pre-defined ontology / ontological commitment]

Nat: I find it interesting that you insist on restricting/removing structural ambiguity, yet leave lexical interpretation completely unrestricted. Isn't that somewhat contradictory?

Form: Not at all. There is an important distinction between a language's *form* and *content*; a CNL should have unambiguous form, yet leave the content completely up to the user, if that's what he wants. Also, remember that although a "formalist" CNL doesn't dictate what words *mean* (that's up to the user), it *does* insist that words are unambiguous (one sense per word). I am perfectly consistent: A CNL should have unambiguous form but unrestricted content.

Also, there is nothing preventing the use of a pre-defined ontology, if one is available, with a "formalist" CNL. If you already have an axiomatized ontology, you can simply plug it in and have the user work with it, providing it obeys the "one sense per word" principle.

Rather, it seems to me that *you* might be the one being self-contradictory! You are happy with ambiguous language but then want to force a pre-defined ontology on the user. What kind of freedom is that?

Nat: Well, a naturalist CNL doesn't have to force a complete ontology on the user, it only needs the parts necessary to represent different disambiguation decisions (e.g., for representing the different meanings of "of"). So there is still flexibility. However, in many cases it's helpful, not harmful, to give the user a pre-defined ontology. Ontology-building is hard, and authoring disambiguation rules harder, and so if we can give both to the user, isn't that a good thing? Otherwise the user has to build it all from scratch!

Form: It seems to me that you are on a slippery slope: If you allow ambiguity then you need disambiguation rules, and disambiguation rules require world knowledge, and world knowledge requires an ontology to express it. So before you know it your CNL is ontology-specific. That's bad!

Nat: That's only partially true as the ontological commitment can be quite minimal, if we mainly use syntactic and lexical knowledge for disambiguation. But look, ontological commitment is generally a helpful thing! Otherwise the user is starting from scratch.

Form: Maybe he *wants* to start from scratch.

Nat: In which case he can use a “naturalist” CNL with the minimal pre-defined ontology.

Form: Or he could just use a “formalist” CNL and have complete freedom!

Nat: but have more work…!

Form: but have less confusion…!

[Practicality and Cost]

Form (continued): In any case, I remain skeptical it's even possible to build a decent "naturalist" CNL that's close to natural language - you are going to need a *lot* of heuristics to make these subtle disambiguation choices correctly. That's a lot of work! Full natural language processing has not made much headway in that direction.

Nat: Yes, but remember the language is constrained, making things a lot easier.

Form: I don’t think that’s enough. Disambiguation is highly domain- and word-specific; you are never going to be successful trying to build a domain-general, “naturalist” CNL. It would be prohibitively expensive!

Nat: Well, it's true that "naturalist" CNLs are probably more expensive to build. But that expense is for a good purpose, namely to make life easier for the users, as the machine is then doing some of the work for them. It seems like money well spent to me. Otherwise, the user will have to do this work himself, carefully crafting his wording to get the output he wants – there’s no free lunch!

There are also several ways to potentially reduce cost: First, modern machine learning techniques can potentially help acquire disambiguation knowledge[5], so it doesn’t all have to be built by hand. Second, we could involve the users in entering the required knowledge via suitable knowledge acquisition tools. And third, remember we don't need "perfect" performance; as we agreed earlier, *any* CNL needs to present its interpretation for the user to check. This failure tolerance removes the need for perfection in the interpreter.

[5] e.g., [20,21]

Form: But why try to make the software smart like this in the first place? I don't think users want smart software, they want controllable and predictable software. CNLs should be as controllable and predictable as a programming language, only more friendly to use.

Nat: No, I would say CNLs should be as natural and easy for the user as normal English, only constrained so that the computer can reliably find the intended interpretation.

[Handling genuine ambiguity, and combining the two]

Form: So what would a "naturalist" CNL do if a sentence is truely ambiguous? e.g.,

(15) A man sees a girl with a telescope.
(16) A lion is a wild animal.
(17) Three men have five cars.

Here you can't rely on heuristics to get the "right" interpretation!

Nat: Well, nothing different; the system makes its best guess and if it's wrong, the user can rephrase the sentence.

Form: Yes, but how does the user know how to rephrase if he doesn't fully understand the interpretation rules?

Nat: Well, the system could offer advice, or the user could rephrase in a less ambiguous way.

You know, there might, indeed, be a case for including some sentence patterns with clearly defined interpretations within the CNL for such eventualities.

Form: This sounds like adding in aspects of a "formalist" CNL!

Nat: Well, maybe some synthesis of the two approaches is the right way forward.

Form: Perhaps!

[Closing]

Nat: Look, I think our opinions originate from rather different philosophical perspectives: Should CNLs be more like an English-like specification language ("formalist"), or more like simplified natural language processing ("naturalist")? And in fact, in implementation terms, there is a continuum between the two extremes - a formalist-like CNL might still include some canonicalization, word-sense disambiguation, etc., or a naturalist-like CNL might include a deterministic core within it. Perhaps no CNL is in fact purely "formalist" or purely "naturalist", and these extremes may not even be fully definable or implementable. Ultimately, what really matters is the practical question: what do users find easiest to understand and work with?

Form: And that may be an empirical question to explore in the future.

Nat: Indeed. Shall we adjourn, and continue over a nice glass of Sicilian wine?

Form: That I can agree to!

Acknowledgements

We are grateful to Vulcan Inc. who supported part of this work through Project Halo, and to Norbert Fuchs for valuable discussions, comments, and organizing the CNL’2009 Workshop itself.

References

1. P. Clark, P. Harrison, T. Jenkins, J. Thompson, R. Wojcik. Acquiring and Using World Knowledge using a Restricted Subset of English. In *Proc. FLAIRS'05*, 2005.
2. A. Pease, W. Murray. An English to Logic Translator for Ontology-Based Knowledge Representation Languages. In *NL Processing and Knowledge Engineering*, 2003.
3. N. E. Fuchs, U. Schwertel, R. Schwitter, Attempto Controlled English. *Proc LOPSTR'98*, 1998.
4. R. Schwitter. English as a Formal Specification Language, *Proc 3rd Proc 13th Int Workshop on Database and Expert Systems Applications (DEXA 2002): W04: Natural Language and Information Systems - NLIS*, Aix-en-Provence, France, pp. 228-232, 2002.
5. J. Sowa. *Common Logic Controlled English.* Technical Report, 2004.
6. V. Tablan et al, User-friendly ontology authoring using a CL. *ISWC'08*. 2008
7. P. Clark, J. Chaw. Capturing and Answering Questions Posed to a Knowledge-Based System. In: *Proc 4th Int Conf on Knowledge Capture (KCap'07)*, 2007.
8. K. Barker, B. Porter, P. Clark. A Library of Generic Concepts for Composing Knowledge Bases. In *Proc 1st Int Conf on Knowledge Capture (K-Cap'01)*, 2001.
9. N. Friedland et al.,. Project Halo: Towards a Digital Aristotle. *AI Magazine* 25 (4), 2004.
10. K. Barker, V. Chaudhri, S. Chaw, P. Clark, J. Fan, D. Israel, S. Mishra, B. Porter, P. Romero, D. Tecuci, P. Yeh. A Question-Answering System for AP Chemistry: Assessing KR&R Technologies. In *Proc 9th International Conf on Knowledge Representation and Reasoning (KR'04)*, 2004.
11. Harrison, P., Maxwell, M. A New Implementation of GPSG. In Proc. 6th Canadian Conf on AI, 1986.
12. Clark, P., Harrison, P. BLUE (Boeing Language Understanding Engine): A Quick Tutorial on How it Works. Working Note 35, http://www.cs.utexas.edu/users/pclark/working_notes/, 2009.
13. K. Barker et al.. "Learning by Reading: A Prototype System". In *Proc. AAAI'07*, 2007.
14. Chaudhri, V., Barker, K., Clark, P. AURA: Automated User-Centered Reasoning and Acquisition System: Phase II Final Report. Technical Report ICS-17540-FR-09, SRI International, 2009.
15. K. Kaljurand. Paraphrasing controlled English texts. In *Proc. CNL’2009 (*CEUR Workshop Proceedings, Vol. 448)
16. H. Tennant, K. Ross, R. Saenz, C. Thompson, J. Miller. Menu-Based Natural Language Understanding. In *Proc SIGCHI Conf on Human Factors in Computing Systems*, 154-160, 1983.
17. T. Kuhn, R. Schwitter. Writing Support for Controlled Natural Languages. in: D. Powers and N. Stockes (eds.), *Proceedings of ALTA 2008*, pp. 46-54, 2008.
18. R. Schwitter, A. Ljungberg, and D. Hood. ECOLE – A Look-ahead Editor for a Controlled Language, In: *Proceedings of EAMT-CLAW03*, 141-150, 2003.
19. Gurevich, O., Crouch, R., King, T., de Paiva, V. Deverbal Nouns in Knowledge Representation. Proc *FLAIRS'06*, 2006.
20. W. Gale, K. Church, D. Yarowsky. A Method for Disambiguating Word Senses in a Large Corpus. In *Computers and the Humanities*, vol 26, pp415-439, 1992.
21. K. Toutanova, A. Haghighi, C. Manning. A Global Joint Model for Semantic Role Labeling. *Computational Linguistics*. 34 (2), pp.161-191. 2008.