A Controlled Language for Knowledge Formulation on the Semantic Web

Doug Skuce

School of Information Technology and Engineering
University of Ottawa
Ottawa, Canada
doug@site.uottawa.ca

Abstract. We present a new type of controlled language, ClearTalk, that attempts to combine the best aspects of both knowledge representation languages and constrained natural language. It offers a flexible degree of formality depending on how much automatic processing is desired, e,g. for translation into a standard KR notation. It requires a few hours training to read and a few days training to write. It is best used with a special search engine but any engine can still access ClearTalk kbs effectively. Knowledge is stored in a structured set of web pages which permit any other web content to be freely intermixed. Over 25,000 statements have been written in ClearTalk; e,g, a 5000 statement kb about Java is effective as a teaching resource. It would be useful in any knowledge formulation situation.

1 Introduction

Formulating knowledge for the semantic web, an inference engine, or natural language processing presently requires that it either be expressed in a formal knowledge representation or written in a constrained form of natural language. The former approaches involve both theoretical complexities and expressiveness limitations that can baffle many potential users and limit applications. On the other hand, the latter, while very user-friendly, leave the writer wondering if and how a particular construct will be interpreted, and have considerable ambiguity in their requirement to look "real". One is left between a rock and a hard place.

Users of KR languages may be unable to express something due to the restrictions of the language, or may not be skilled or willing enough to master the intricacies of the language. Controlled languages do not suffer expressiveness limitations; large organizations can successfully produce masses of complete documentation. But such documents cannot be automatically converted into formal knowledge bases.

Hence the question: *is their some hybrid trade-off between these extremes where we can have both expressiveness and yet still retain a fair amount of exactness?* To answer in the affirmative, we have designed a *knowledge formulation* language, *ClearTalk*, (CT) that 1) "looks real", 2) has adequate expressiveness, and 3) can be translated into a formalism if required. We have observed that, for applications not re-

requiring a formal translation (and there are many), the language will be more usable, i.e. used, if it is not too formal, implying some ambiguity. CT is designed so that an author can chose to leave or remove ambiguity, depending on the need. For human processing, ambiguity is usually not a problem. In fact, people generally prefer naturalness to exactness, unless exactness in essential. Using optional rules that add formality if required, translation of CT to a KR representation could be automatic in most cases. Hence CT offers optional specification of 1) syntactic structure; 2) conceptual relations; 3) word senses.

We have found it useful to distinguish two types of applications: 1) knowledge transfer between *people*, e.g. in teaching or in any reference document, and 2) knowledge transfer from people to a *KR system.* We are mainly interested in the former, but CT is still highly applicable in the latter. Here one could first prepare knowledge in CT and then translate it into a KR formalism, which is not designed for human friendliness, with little human intervention. The situation we are most interested in is large document collections such as the web, where one seeks facts that lie buried in some document if only some intelligent retrieval system could find them. Attempts to do this, chronicled in the TREC series [1], are often unsuccessful due to the complexity of *unconstrained natural language* (unl) and the lack of knowledge-based reasoning.

CT must be able to express *all* the knowledge found in the unl of technical texts, while eliminating most types of syntactic ambiguities that thwart natural language processing. It has been used over the past five years to encode over 25,000 facts in four technical domains: astrophysics, software engineering, Java, and a description of itself. Each domain has been formulated into a CT kb, first using a system we built in the nineties, based on an RDBMS, and currently using a new system we are developing, based on indexed web pages. One of its uses has been to facilitate teaching of Java concepts (see demo of year 2000 system at www.factguru.com; current system not ready for demo). A prime design goal was to leave no important fact unexpressed, yet we believe CT remains quite simple. Hence we now feel ready to design CT processors that will not have to deal with all the problems of unl (e.g. a parser/checker using the GOLD parser-generator). Storing CT in a kb to facilitate web knowledge retrieval, a straightforward matter on which we are now working, could make finding detailed facts infinitely easier than with today's unl web pages. Instead of asking for pages containing words (i.e. what today's search engines do), in our system one asks to find sentences expressing certain semantic types of facts about certain concepts.

CT is designed to enable people who do not understand formalisms to easily but precisely express their knowledge without running into limitations that would frustrate them. In its present form it looks like English, but any other natural language could be substituted. Machine translation or search engine fact retrieval (discussed in Section 4) of CT would be virtually perfect. CT can be understood by any English speaker with almost no training; writing it requires learning about one hundred rules, many of which we present in Section 3.

This paper will continue with a discussion of related work, followed in Section 3 by the description of CT *given in CT itself.* We do this for two reasons: 1) to provide examples of CT; 2) to illustrate one of the tasks for which it is designed: CT serves as

a human-human as well as a human-machine medium. In Section 4, we shall describe how CT can be processed with a specially-designed search engine, an application that makes CT-based knowledge far more widely usable than conventional formal knowledge bases. Finally, Section 5 discusses some critical questions and offers some suggestions for further work.

2 Related Work

The following research areas have most contributed to our ideas: *controlled languages, AI knowledge representations,* and the *semantic web.*

Controlled languages. Most CLs are simple subsets of natural language constrained to facilitate machine processing, often for machine translation (e.g. AECMA [2]). These are used in conjunction with a checker that flags syntax, style or lexical violations. Writers learn to use them by learning mainly what *not* to write. CLs have had considerable success in large corporations (e.g. Boeing, GM, Caterpillar) where they are used to produce user documentation, often for machine translation systems [3].

The long-standing project at Carnegie Mellon University to use a controlled language (Kant Controlled English) for MT is particularly germaine. Recently, they have added a new feature to their system: to be able to convert its internal representation of the input text into conceptual graphs or OWL, from which they can do deductive question answering. They make it very clear though that they cannot process unl text directly: it must be rewritten sentence-by-sentence into their controlled language. The examples they give in fact conform to the rules of CT [4].

There have been only a few proposals for more formally prescribed languages that look "natural" such as CT, the origins of which lie in the author's PhD dissertation [5]. For example, Web-KB [6] is a version of conceptual graphs for which an "English-like" syntax is available, intended for web-based knowledge formulation. But one is not writing English, one is writing cgs. Similarly, Attempto Controlled English [7] is an English-like syntax for first-order logic, but again one must understand fol to use it correctly. Only small kbs (a few hundred axioms) have been expressed in ACE, a limitation of theorem-prover technology, which does not scale up.

Formal knowledge representations Any of the classic KR languages (e.g. KIF, CycL, cgs, description logics) could be a target for translation of CT. While knowledge formulation has often been done directly in these languages, they are a long way from natural language in their appearance and hence ease of use, even by seasoned researchers. (If you find reading our description of CT in Section 4 easier than reading a standard KR language, then our point is proved.) However, while designing CT, we constantly asked ourselves "how would this translate into a classical KR?" (We used mainly cgs.) We believe that CT can be completely translated in a KR such as cgs or its equivalent, and are now designing a translator.

Semantic web Various mark-up methods have been proposed, based almost totally on XML as an underlying syntax. The most ambitious of these are oriented toward AI-style inferencing, (e.g. the OWL set of languages [8]). Those that aren't rely on

other more task-specific forms of processing (e.g. Dublin Core [9]). Being XML, none of these notations is human-friendly, though they are read/writeable with patience.

Nearly all AI-style semantic web proposals that have been developed are extensions of frames+rules or description logic foundations (e.g. Ontobroker [10] or OWL), and are based on binary relations. We have found however that, for high expresiveness, one needs a "high-end" (read: intractable) knowledge representation, of which cgs are arguably the best example, being motivated specifically as a KR for natural language. [11] Recently semantic web researchers have begun to look at these more complete formalisms such as conceptual graphs (e.g. [6], [12], [13]).

3 ClearTalk Overview

This section will introduce the main features of CT, using CT itself. Due to space limitations, less than half of the CT rules can be given, and we also do not provide much syntactic detail. Often we will have to use some feature before it is explained, or without explaining it at all. We believe that CT is sufficiently friendly that this should not cause a problem. CT statements are distinguished by a different font (we use Arial) and ended by delimiters (we use *period newline* or *period* |).

3.1 ClearTalk Documents and Knowledge Bases

ClearTalk documents –(

- look almost like natural language.
- can be converted to KR formalisms almost automatically.
- can be almost automatically translated to other natural languages.
- usually contain (both CT and ordinary language).
- are normally authored in a (word processor or HTML editor).
- are HTML documents.
- can easily be converted to PowerPoint
- are usually accessed as Web documents using (direct access or a special search engine).

).

ClearTalk knowledge bases –(

- are a set of web pages in specified formats.
- include a lexicon.
- include various hierarchies. *example:* kind of.
- can be searched by (ordinary and special) search engines.

).

3.2 CT Statements

CT statements --(

- are written with (a distinct font and certain delimiters). reason: to distinguish them from ordinary language (aka: natural language). *comment:* the default font is Arial.
- use a small set of keywords.
- have a very simple syntax that can be made unambiguous. *comment:* The only syntactic ambiguity is phrase attachment. Syntactic ambiguity can be removed by using parentheses.

).

3.2.1 Statement Syntax

CT statement syntax: basic statement || compound statement || special statement.
comment: || *means*: choice.

Basic statement syntax:
subject [predicate] [complement] [modifier phrases] [('.' || ';') newline keyword phrases]. ('.' newline || '.|' || '.' whitespace '|');

comment: a period is required before a keyword phrase if a statement has a predicate. *purpose*: to improve readability;
comment: the statement terminator is a period followed by a (newline or vertical bar);
comment: modifier phrases modify a preceding (verb or noun), but they can be ambiguous without syntactic information;
comment: modifier phrases terminate with semicolon or period;
comment: keyword phrases *purpose*: to supply information in certain well-known semantic relationships; they may be in natural language.
(Compound statements and special statements) are discussed below.

3.3 Noun phrases

a noun phrase –(

- is a kind of (proper noun, count noun, mass noun, pronoun or variable).
- can be modified by adjectives if it is a (count or mass) noun.
- is (either singular or plural) if it is a count noun.
- can be modified by a quantifier if it is a (count or mass) noun. *examples*: the boy, 3 balls, few women, some water.
- can be modified by 1 or more modifier phrases. *examples*: the cat on the mat near the door ate the mouse. *comment:* here is an example of prepositional phrase ambiguity: does "near the door" modify "cat" or "mat"? By default, it modifies the subject, 'cat', but to have 'near the door' modify 'mat we can write (the cat on the (mat near the door)). | Modifiers that follow the verb modify the verb by default.
- can be modified by 1 relative clause. *examples*: (the man who came today; the man whose shirt is white; the box that is on the table)
- can be modified by 1 qualifying phrase. *example:* ClearTalk (aka: CT)). *comment:* the qualifying phrase must follow immediately in parentheses.

).

The quantifier (= 'any', 'each' 'every' or 'all') *means*: universal.
The quantifier (= 'a') on a subject *means*: universal.

Non-subject *means*: a noun phrase that is not the subject.
The quantifier (='a') on a non-subject *means*: 1.
A quantifier is not universal iff it is existentially dependent on the subject.
The following are instances of quantifiers: (no, some, many, a few, 0 or more (aka: 0m), 1m, most, <int>, less than <int>, from <int> to <int>). *comment:* any others that behave like these would be permitted.
A qualifying phrase begins with (one of: 'aka:', 'e.g.:', 'i.e.':, 'a', 'called:', '=' a (noun or variable), a variable) *examples*: Microsoft (a: company), ClearTalk (aka: CT), a person (P).
aka *means*: also known as. | aka *purpose*: to introduce a synonym.

3.3.1 Pronouns and references

A pronoun may be used later in a statement if the referent that it refers to can be determined by using simple rules. *examples*: (Jane called her friend; skiing is fun but it can be dangerous; John brought some boxes but they were too small).

('it', 'itself', 'they' and 'themselves') refer only to the subject.

To refer to some noun other than the subject you may use (a pronoun or 'the' or 'that') followed by the noun. *example:* A variable that refers to an object contains a pointer to that object.

3.3.2 Variables

A variable may be (introduced in a qualifying phrase and used later in place of a pronoun). *example*: all students who are enrolled in a course (C) must have the prerequisite for C.

3.3.3 Relative clauses

a relative clause --(

- begins with ('that', 'which', 'who', 'whose' or 'what').
- modifies the noun that it follows.
- contains a predicate.
- may contain (a complement and modifiers).

).

Relative clause *examples*: (

The type of a variable determines the kind of objects that the variable may contain;
The person whose car is outside must move the car).

3.4 Embedded statements.

Sometimes a statement must be part of a larger statement, for example, when using verbs referring to mental acts. Such statements must be enclosed in [- -].

An embedded statement –(

- is part of a statement.
- is delimited by brackets (= '[-' '-]').
- *example:* Mary hopes that [- Bill loves her -].

).

3.5 Conjunctive phrases

A conjunctive phrase –(

- conjoins (nouns, verbs, adjectives or adverbs).
- is constructed with 1 or more conjunctive phrase connectors.
- requires parentheses. *example:* (both the boy and the girl).
- may contain a conjunctive phrase. *example:* (both Bill, (either Mary or Jane)).
  examples: (all of: red, green, blue), (red but not pink).
- *comment:* note that conjunctive can mean AND or OR.
).

The following are instances of conjunctive phrase connectors: (none of, 1 of, 1 or more of (aka: 1m of --), both – and --, some of --, all of --, -- but not --, -- or -- , either – or -); *comment*: each conjunctive phrase connector can be expanded into a full compound statement.
A conjunctive phrase is equivalent to two or more statements. *example:* Bob likes (both Mary and Jane) *means*: Bob likes Mary and Bob likes Jane.

Parentheses must be used around conjunctive phrases. *reason*: to eliminate syntactic ambiguity.

3.6 Predicates

A predicate --(

- contains a verb phrase with 0 or more adverb modifiers. *example:* usually does not eat quickly.
- may contain a noun-preposition pair that names a relationship if the verb = ('is', 'are' or 'has'). *examples*: is the father of, is equivalent to, has as color.
- may combine a passive verb with a preposition. *example:* is connected to.

).

3.6.1 Verb phrases

A verb phrase --(

- can contain a particle. *examples*: look up, skip over
- can contain a conjunction of verbs. *example:* (eats and drinks)
- can contain auxiliary verbs. *examples*: (was going; has been eating)
- can contain (modals and negation). *example:* must not go
- can contain adverbs. *example:* (eats quickly; frequently stops)

).

3.7 Complements

A complement --(

- is usually a noun phrase. *example:* Bob loves Mary.
- can be an infinitive phrase. *example:* Bob wants to marry Mary.
- can be an adjective that modifies the subject if the verb is an instance of 'to be'. *example:* Mary is pretty.

).

3.8 Modifier phrases

Modifier phrases --(

- follow a (noun or verb) that they modify. *example:* Bill bought a car while shopping. *comment:* 'while shopping' modifies 'bought'.
- modify the (subject or verb) by default.
- are usually prepositional phrases that contain noun phrases.
- begin with a word that acts like a (preposition or quantifier or verb-modifying adverb or a conceptual relation keyword).
- end with a (head noun or adjective).
- are not explicitly delimited unless required.
- are subject to possible syntactic ambiguity unless they are delimited by parentheses. *reason*: they may not immediately follow the word that they modify.). *comment*: examples are given below.

3.9 Specifying conceptual relations

If you wish to explicitly specify binary conceptual relations (e.g. agent, theme, source) then you can use a notation in a manner that is similar to conceptual graphs with the predicate as the initial part. *example:* go – (agent: John) (dest: Boston) (instr: a bus). *comment:* you may want to compare CT with the standard linear cg form.

3.10 Keyword phrases

The final component of a statement may be 1 or more keyword phases. | A keyword phrase begins with a keyword that ends in a colon. *purpose*: to add information to a (word or statement).

Each keyword has a domain-independent semantics. One could create domain-specific keywords.

keywords instances: (

- reason/why: *purpose*: gives a reason;
- purpose: *purpose*: gives a purpose;
- causes/caused by: *purpose*: gives a causal relation;
- except: *purpose*: lists exceptions;
- definition: *purpose*: gives the definition;
- example*:* *purpose*: gives examples;
- means: *purpose*: says what some phrase means;
- parts: *purpose*: lists parts;
- kinds: *purpose*: lists kinds (i.e. subconcepts);
- instances: *purpose*: lists instances;
- syntax: *purpose*: gives syntax for language constructs;
- code: *purpose*: gives code examples;
- where: *purpose*: to define words or symbols used previously;
- how: *purpose*: to state how to do something;
- when: *purpose*: to state a (time or temporal) relation;
- followed by: *purpose*: indicates a syntactic sequence

). *comment:* these are about 1/2 of the keywords.

3.11 Adverbs

Adverbs can only modify (verbs and adjectives). Any adverb that modifies a verb must be adjacent to (that verb or another adverb).

3.12 The pronoun 'you'

If a statement has subject (= 'you') then it's purpose is to tell you (what you may do or how you may do something). | To say how to do something you use a special statement that has (an infinitive or 'you') *example:* to bake a cake you follow a recipe.

3.13 Statement levels

'o-'*n* may be placed at the end of a CT statement. where: n represents a digit indicating an importance level and if n=1 then this statement has prime importance. *means*: the statement should be learned before statements of lesser importance. | Statement levels *purpose*: to permit summarization. *how*: by showing only statements above a certain level. *example*: an important statement should be read before other statements o-1.

3.14 Special Statements and Predicates

Special statements are statements that have certain special predicates that serve a well-defined (axiomatizable) semantic purpose.

The following are instances of special predicates: (

- is a kind of
- is an instance of *comment:* the phrase “is a” is never used without (‘kind’ or ‘instance’). *reason*: to avoid ambiguity.
- is different from
- is the same as || is equal to || =
- is part of
- has as part || has as parts
- is disjoint from || is distinct from
- is also known as || is a synonym of) *comment* : (the plural and negated) form of each of the above is permitted

).

3.15 Classifying (Special) Statements

There are 4 kinds of classifying statement: --(

- the *kinds-of* classifying statement.
- the *instances-of* classifying statement.
- the *following-are-kinds-of* classifying statement.
- the *following-are-instances-of* classifying statement.

).

kinds-of classifying statement *syntax*: There are <n> kinds of *noun phrase* [based on *noun phrase*]: (*list of noun phrases*). *means*: these are a disjoint partition based on some attribute. *comment:* the optional ‘based on’ phrase can name the attribute that produces the distinction. *example:* there are 4 kinds of people based on age: (infants, children, teens, adults).

instances-of classifying statement *syntax*: There are <n> instances of *noun phrase: means*: these are all the instances that exist. *example:* there are 10 instances of digits: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

The “following” form of classifying statements provide non-exhaustive lists.

3.16 Compound Statements

a compound statement --(

- may be constructed using 1m statement combiners.
- may be constructed using the bullet construction for multiple statements that have the same initial words. *example:* this statement uses the bullet construction.

).

3.17 Statement combiners

The following are instances of statement combiners: (';', and, or, but, but not, if, if -- then --, if and only if (aka: iff), if --- then –, (and conversely || vice-versa) (*means*: iff), unless, whenever, when, while);
comment: any word that (behaves as a statement connector and has a clear meaning can be used);
comment: if more than one connector is used, parentheses must be used to show precedence:
example: if (a person (X) knows a person (Y) and Y knows a person (Z)) then X can meet Z

3.18 The bullet construction

The bullet construction is a convenient syntactic sugar for multiple statements that have the same initial part; it is expanded by replacing each bullet in a statement by the global initial part that is given on the first line before the --; it has been used above.

The initial part of the bullet construction is normally the subject but it may include the predicate. *example:*

my cat ate --(
- 1 mouse.
- at least 3 rats.
- my homework

).

3.19 Temporal keywords

When a temporal relation is meant, you must use a temporal keyword to connect statements; it cannot be described by a conjunction.
The following are instances of a temporal keyword: (before , after, when, while , whenever, and then, next). *example:* Bob bought a car after he got a job.

3.20 Terms and the Lexicon

A term is a noun, verb, adjective, or adverb that has a fixed specific meaning in some domain. It can be made up of several words. All terms are entered in the lexicon and given a definition or *purpose*, and hopefully a genus (super). If a term has several senses, they are distinguished by appending sense numbers after a dash. For work on the semantic web, lexicons would be namespaces designated by urls. Our current lexicon consists of a web page using keywords in a fixed format. The lexicon defines an ontology. (Exercise: this paragraph is almost CT. Try rewriting it.).

lexicon entry example:

car is a kind of: motor vehicle;
pos: count noun;

fact page: <link to page of car facts>;
French: voiture (f)

3.21 The verb "to be"

The verb "to be" behaves specially. Any noun, preposition, adverb or adjective that directly follows it is almost always considered part of the predicate. However the verb "to be" also occurs in some of the special statements, where its behavior is more specific. (See Special Statements). In the following, the predicate is in italics:

Java *is very popular*.

Source code *is kept in* files.

John *is the father of* Mary.

3.22 "How to" with 'you' and 'how:'

You can use the pronoun (= 'you') to give "how to" instructions. *example*:
You define inheritance in Java by creating an inheritance hierarchy. *how*: by using 'extends.

The basic pattern is 'you' verb1 modifiers1 'by' 'verb2 modifiers2'. An equivalent form is 'To' verb2 modifiers2 'you' verb1 modifiers1.

An *informal procedure* involves giving "you" a number of steps, as in the steps for baking a cake:

To bake a cake --(
- you get the ingredients.
- and then you mix the ingredients together in a bowl to make the batter.
- and then you dump the batter into a pan.
- and then you bake the batter in the pan in an oven for several hours

).

3.23 Parsed examples:

Each subsequent example will be parsed using parentheses. *purpose*: to illustrate the explicit syntax of CT when it must be specified.

- s will denote the subject.

- p will denote the predicate.
- v will denote the verb (part of the predicate).
- c will denote the complement.
- m will denote modifiers;
- k will denote a keyword phrase;
- *comment*: these 5 symbols do not appear in normal CT. They are being used here for explanatory purposes.

Here we show several variations of modifier phrase attachment.

> (s: The cat) (on the mat with the hole) (p: ate) ((c: the fish) (m: on the plate)).

The hole is in the mat, not the cat, which is on the mat. *comment*: 'on the plate modifies 'the fish'. Without the parentheses, it would by default modify the verb which is probably not what is intended.

> (s: Bill) ((p: usually dates) (c: Mary) (m: on weekends.)) (because: (s: he) (v: is) (c: busy)).

'Because' is a statement connector that attaches the second statement to the first. "He" refers to the subject. Bracketing 'on weekends' with 'Mary' would not make sense.

> (s: Java) ((p: was developed) (m: at Sun Microsystems) (m: in the mid-nineties)).

Both modifiers modify the predicate. Note that 'in the mid-nineties' is not a location but a time period and hence could not modify Sun Microsystems, but most parsers are not that smart. In this example, the default is what is desired.

(s: All Java compilers) ((p: compile) (c: source code) (m: into files (m: that (v: contain) (c: bytecode)))).

This example shows how even a simple syntactic structure can be awkward to specify formally.

((s: A Java VM) (p: will not allow) (c: violations (m: of certain security constraints))) when ((s: applets) (p: are downloaded) (m: over the Internet.)).

A connector like 'when' or 'because' connects two or more statements:

((s: You) (p: do not need) (c: to free (c: objects) (m: from memory))) when ((s: you) (p: no longer need) (c: these objects). (unlike: (C and C++)).

(s: Java) (p: is less efficient than) (c: (C and C++)) because: (
(s: Java's (safety checks and garbage collection)) (p: slow down) (c: execution)) and (((s: the interpretation) (m: of its bytecode)) (p: is not as fast as) (c: direct execution (m: of machine code))).

Note that the parentheses around ‘safety checks’ and ‘garbage collection’ are always required in the unparsed CT (enclosing any conjunctive phrase) to avoid a common cause of syntactic ambiguity. To expand the ‘and’, write the statement twice, once with ‘C’ and once with ‘C++’ and connect them with ‘and’.

The following illustrates how to specify syntax and give a code example.

(s: Java class declaration) (k: *syntax*: ‘class’ *className classBodyBlock*)
comment: The syntax keyword *purpose*: to specify BNF:
comment: You may use statements without verbs for some keywords.

the credit method (a method) *code*:

```
public double credit(double amountToCredit)
   {
  balance = balance + amountToCredit;
  return balance;
   } .
```

4 Searching CT and the CT Knowledge Base

A traditional search engine is given a set of search terms and returns a ranked list of files or web pages. In some, the hit contexts are highlighted, and one may jump from one to the next. A few, such as the dtSearch engine [14], can output windows such as paragraphs that surround hits, but finding sentences is too difficult. Because CT is well controlled syntactically and has keywords, it becomes possible to implement a search engine that can find things a normal search engine cannot. We have done this, using dtSearch as our basic engine, with a Java servlet architecture using Apache. Here are some types of queries that we can or will be able to make:

- show all occurrences of the term t with window size ±w statements (i.e. a basic search engine function)
- ditto, but ordered in a useful way, such as alphabetically by subject, or by verb.
- ditto, but *summarized* by showing only levels <n.
- ditto, but seeking certain keywords, such as definition, cause, or reason.
- ditto, but requiring a second word to be present, or one that has a certain type of meaning, such as a causal verb like “create” (this latter would require a lexical resource such as Wordnet [15] which would be straightforward to incorporate.)
- show all facts about *part time graduate students who are taking courses in this semester (a university calendar)*
- *how do I print a file to a networked printer in Windows XP*? (a new kind of help system)

Briefly, A CT kb consists of three elements: 1) linearly organized documents such as this one, intended to be read in the normal manner, and containing CT interspersed with ordinary language and any other material. These serve as normal introductory or advanced material. 2) The lexicon, containing all terms and their lexical information such as definitions, super-ordinate concepts, etc. All terms hyperlink to the lexicon, and its entries link to the "topic: pages. 3) the "topic" pages, each of which contains all facts where one topic is the syntactic subject, usually sorted by the predicate. These are not meant to be read linearly unless one wishes to learn everything possible about the topic. All are indexed by our special search engine which permits finding facts with particular semantic structure. All documents at present are HTML, but they could be XML if this would add some advantage, e.g. if we stored parsed versions of the facts. More likely, they could be converted into some other representation which might be XML-based for further semantic processing.

The kb is intended to be shared along the lines of the Ontoweb project [16]. People with suitable permission may add knowledge either as new statements on existing pages or new pages, consistent with the ontology as defined in a shared lexicon. Note that these pages can be anywhere on the web, as long as the indexer can find them. We intend to pursue this technique as a medium for scientific publishing, e.g. if all Java knowledge were in such a format, it would be much easier to find and understand.

5 Discussion

Will people use it? Probably the most critical question is "will people be willing to make the effort to author documents in CT?" We believe that, when the advantages become clear, and CT assistants (software) become available along with CT search engines, the answer is yes. Consider similar examples that are very labor-intensive (often machine-assisted): translation of unl, preparing web pages, writing music, preparing XML pages, writing in any kind of formalized specification language, including all "KR" languages; and programming itself, a much more difficult task. In our group, three others besides myself have written CT.

How does CT relate to proposals like DocBook? DocBook [17] is an SGML/XML standard for technical documents. In addition to "large" structure, it provides many tags reminiscent of our keywords. But it has no notion of linguistic structure, e.g. no "subject" or "predicate" tags. So CT may be seen as an extension of DocBook that incorporates elements needed for linguistic, retrieval, and KR processing.

Does CT help students learn? To properly conduct an experiment to demonstrate this would be very difficult in a university environment. However, we did perform an informal test. Eighty students were divided into two groups and given forty-five minutes to answer questions about applets, a new topic. Each had their textbooks but the test group had access to the kb. It got a 50% higher mark. This shows that they could find facts faster, which is a prime function of the kb. However some of the difference might be due to better understanding. Students are taught to try to learn all facts about one topic at a time, which are all available either on the topic pages or from the search engine, which locates facts about X not stored on the page for X.

Future work. We hope to have our special search engine running by fall 2003, and plan next to work on a parser/checker as described above, i.e. a CT authoring system with features that extend those of today's CL checkers. Any words that were not in the lexicon, or that were ambiguous, would be flagged, and the user could take corrective action. Similarly, grammatical errors would be flagged, though CT is considerably more strict than most CL checkers, many of which do not perform a complete parse. The system should at least identify the subject, predicate, and complement, which could be tagged for latter use such as in the searches described above.

References

1. TREC conferences. http://trec.nist.gov/
2. AECMA http://www.aecma.org/Publications.htm
3. Proc. Third International Workshop on Controlled Languages Applications, Seattle, 2002.
4. Nyberg, E., Mitamura, T., Baker, K., Svboda, D., Peterson, B., and Williams, J. Proc. Deriving Semantic Knowledge from Descriptive Texts using an MT System. Amer. Mach. Trans. Assoc., 2002.
5. Skuce, D. Towards Communicating Qualitative Knowledge Between Scientists and Machines. PhD Dissertation, Dept. of Electrical Engineering, McGill University, Montreal, 1977.
6. http://meganesia.int.gu.edu.au/~phmartin/
7. http://www.ifi.unizh.ch/groups/req/ftp/papers/ACE_Manual.pdf
8. http://www.w3.org/2001/sw/WebOnt/
9. http://dublincore.org/
10. http://ontobroker.aifb.uni-karlsruhe.de/content.html
11. Rassinoux, Anne-Marie, Robert H. Baud, Christian Lovis, Judith C. Wagner, Jean-Raoul Scherrer (1998) "Tuning up conceptual graph representation for multilingual natural language processing in medicine," in M-L Mugnier & M. Chein, eds. (1998) *Conceptual Structures: Theory, Tools, and Applications*, Lecture Notes in AI 1453, Springer-Verlag, Berlin, pp. 390-397.
12. http://www.w3.org/DesignIssues/CG.html)
13. Gerbé, O. and Mineau, G. The CG Formalism as an Ontolingua for Web-Oriented Representation Languages Uta Priss, Dan Corbett, Galia Angelova (Eds.): Conceptual Structures: Integration and Interfaces, 10th International Conference on Conceptual Structures, ICCS 2002, Proceedings. Lecture Notes in Computer Science 2393 Springer 2002

14. http://www.dtsearch.com/
15. http://www.cogsci.princeton.edu/~wn/
16. http://ontoweb.aifb.uni-karlsruhe.de/
17. http://www.docbook.org/