Accepted: AAAI Workshop on Knowledge Representation Aspects of Knowledge Acquisition, San Jose, July 1992

Informality in Knowledge Exchange

Timothy C. Lethbridge
Doug Skuce

Department of Computer Science
University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
(613) 564-8155 tcl@csi.uottawa.ca
(613) 564-4518 doug@csi.uottawa.ca

Abstract

We believe that a knowledge exchange language (KEL) should be able to support informality. By this we mean the presence of knowledge structures for which the KEL definition specifies no semantics. Such structures are intended to be interpretable only by humans, or by specialized programs into which a KEL file is loaded.

We explain the rationale for this assertion and point out some shortcomings of current proposals. We describe several types of informal knowledge and suggest how such knowledge can be handled in a KEL. CODE4 is used as an example of a working knowledge management system whose CKB file format has some of the capabilities we propose.

1. Introduction

There are active proposals for the adoption of a knowledge exchange language (KEL) that would allow disparate systems to share knowledge bases. Among the suggestions are the Knowledge Interchange Format (KIF)[1] by Genesereth and Fikes [GENES 91]; and Sowa's conceptual graphs [SOWA 84].

Most of the debate about KEL proposals has centred on their syntax and formal semantics[2]. Little or no attention has been given to the exchange of knowledge structures that are 1) only partially formal, or 2) differently interpretable by various subsets of the exchanging systems. We term such structures *informal*.

Currently most knowledge representation languages contain some kind of informality, although this is often not properly appreciated. The following are examples: The mere use of English words as logical symbols results in knowledge bases whose 'meaning' is largely grounded in humans' interpretations of the symbols (e.g. predicate names that are English words). Most KR languages allow some kind of "commentary" knowledge to be integrated. Other KR languages contain primitive elements whose formal semantics are given by an implementation language such as Lisp. We will argue that these latter elements should be considered informal too, *relative to* a KEL[3].

In this paper, we first expand on what we mean by informal knowledge and then explain the importance of being able to represent it. Finally we propose how informality might be handled in a KEL.

Our ideas are being implemented as part of CODE4 [SKUC 92], a generic knowledge representation environment that supports informality and usability in preference to strong inference capabilities. CODE4 is derived from earlier work on CODE2 [SKUC 91]. CODE4 supports various knowledge representation schemata[4]:

---

1 The term KIF has developed two senses: The specific proposed KIF language, and, more generically, any proposal for some future standard language whose requirements are still being actively researched and debated. We suggest the replacement term KEL for the generic sense.

2 Examples of formal semantics can be found in: [SOWA 84] for conceptual graphs, and [SCHMO 91] for the KL-ONE derivative, NIKL.

3 Lisp the *language*, is undoubtedly formal; however, a piece of Lisp embedded in some knowledge base would be considered informal relative to the KR language's formal semantics if those semantics do not contain the Lisp formal semantics. We expand further in section 2.

4 The terms *schema*, *representation*, *language* and *format* are often used interchangably (with considerable resulting confusion). We use *schema* as the most general of these terms. We use *representation* (when it does not qualify something) to mean, 'an instantiation (e.g. as text or data structures) of a schema'. *Format* and *language* are reserved for schemata whose instantiations are in a linear or planar form. Some of our terminology is due to Bradshaw et al [BRAD 91].

• A *conceptual schema*, that is abstract and mathematical.

• Several *user interface presentation schemata* for the display of knowledge in an easy-to-understand format (instantiated as *mediating representations*).

• An *internal physical schema* for efficient inferencing and manipulation (instantiated as the data structure arrangement of the current implementation).

• An external compact file format for the exchange of knowledge bases.

All these representation schemata have fundamental differences, but all handle informality. It is the latter, the *CODE4 Knowledge Base* (CKB) file format, that has many properties we hope will be possessed by a standard KEL. A partial syntax for the CKB format is given in the appendix.

If a satisfactory standard KEL can be agreed upon, we would look forward to using it instead of CKB format so that we can exchange knowledge, both formal and informal, with systems as diverse as CYC [LENA 90] and Classic [BRACH 91]

2. Towards a Definition of Informal Knowledge

During the first Workshop on Informal Computing [MUND 91], attempts were made to define informal knowledge (or its converse, formal knowledge). The following points are derived in part from the discussions at the workshop[5]:

a) Whereas purely formal knowledge has a semantics,. i.e. its syntax is mapped to some underlying model (e.g. set theory), informal knowledge lacks such a mapping. Informal knowledge can only have meaning by being 'grounded[6]' in external interpretation, e.g. in a human.

b) The formality of a representation is *relative to* the declared semantics of a representation schema; i.e. a representation's formality is not an absolute attribute. We can consider knowledge in the context of different knowledge representation schemas (e.g. as it is physically moved, or as different processes operate on it), thus knowledge that is formal under one semantics may be informal under some other semantics, and vice versa. For example, the expression `(+ 2 2)` may be considered formal Lisp, or an informal character string, or informal marks on a piece of paper.

c) The richness of the set of computable relationships among the knowledge elements (i.e. the number of deducible facts normalized with respect to the KB size) depends on the formality, since the semantics determine what is computable. The richness of the relationships therefore might lead to a way to measure relative formality[7].

This paper will focus on points a) and b). Different systems that are to exchange knowledge will inevitably have different semantics for their individual conceptual schemata. It is reasonable to assume that even after thorough rationalization, it will not be possible to map the entirety of any given system's semantics into those of a common schema whose instantiations are interpretable by all other systems. In fact, as we shall point out in the next section, it is very important to give researchers the freedom to extend their semantics at will, as long as a way is found whereby such systems can remain compliant with a KEL standard.

A KEL standard should therefore have a very small core with a well-defined semantics to which *every* system can map (e.g. one based on first order logic), as well as a syntactic mechanism for handling elements that are informal relative to the KEL.

The following is a (perhaps partial) list of knowledge representation elements which should be considered informal since they have a syntax but no formal semantics:

i. Symbols:

Here we include things like conventional constant symbols in logic that are usually represented as character strings having some meaning to humans, and perhaps to specialized computational subsystems.

---

5 We do not intend to imply that these points were agreed to at the workshop. For example, there was debate about whether point a or point c in fact leads to a definition of informality. Here, we have concluded that interconnectedness is a *consequent* of formality, rather than a defining property.

6 We use the word *ground* in the general sense of 'making a connection to'. In logic, one grounds a variable by making a connection to some constant. Here, one grounds a symbol by making a connection to some human (or machine) so that the next time the human (or machine) sees the symbol the same neural pathways (or procedures) are activated. The same symbol can be grounded differently in different contexts (e.g. two humans, or the same person at different times, examining a knowledge base will likely make different interpretations).

7 Measuring relative formality or informality is a subject of ongoing research for the authors.

We exclude syntactic markers from our definition of symbols, since their identity affects syntax and hence semantics.

Under the core semantics of a KEL it should always be possible to substitute all occurrences of a given symbol by some other, such that the formal meaning of the knowledge base would be unchanged *relative to the core semantics*. Of course in such a case, there is likely to be a knowledge change due to the semantics of the symbol itself as interpreted by (i.e. relative to the semantics of) a human or a computer system.

We should not restrict our definition of symbols to include only character strings, although it would be convenient if all symbols in a KEL were constituted of characters. In CODE4, we allow symbols to be arbitrary images (pixel maps), but these are converted to a compressed string format for saving to a file.

ii. The ordering, arrangement or visual appearance of knowledge elements:

In pure logic, the ordering of statements has no relevance. From the point of view of a human looking at a knowledge structure, however, the only way to distinguish elements might be how they are ordered or arranged in space. Alternately, elements might only be distinguished by their color, font etc. For human interpretation, at least, attributes such as these can be extremely important. For this reason, in CODE4, we can currently save some such information to our CKB format. In future we plan to extend this.

iii. Unconstrained values (typically strings) entered in slot values or as predicate arguments.

There has been a tradition in frame systems to associate arbitrary Lisp functions with slot values. This is often required for complex applications. Since different KR systems use different languages (or at least have different libraries of executable code available), the use of such arbitrary procedural attachment must be considered fundamentally informal relative to a KEL, i.e. my Lisp may not be translatable to your Prolog.

It is, however, useful to allow *any expression* to be entered as a slot value in order to cater to the needs of informal knowledge acquisition. See section 3a).

iv. Comments and annotations of any sort:

Annotating knowledge structures is allowed in most languages and is explicitly informal. We would not expect to give a formal semantics to such structures. However, we would suggest as a guideline that comments only use a circumscribed terminology consistent with the symbols.

All knowledge representation languages known to the authors allow some informality, e.g. they use symbols that are English words. We believe that such languages should have explicitly identified types of informal elements, such as the above, and allow all of them if possible[8].

Using metaknowledge (higher order knowledge) the four types of informal elements listed above could all be expressed in a syntactically uniform manner. For example in a predicate-based schema, ordering might be expressed using the predicate:

```
position (concept1, 3).
```

We do not recommend this approach, preferring to keep the types of formality syntactically distinct. The reason is that amount of such metaknowledge would tend to dominate, hence we should use syntactic shortcuts. Of course, a system that interprets the KEL could convert the syntactically-distinct types of informal knowledge into metaknowledge for its own internal knowledge representation

3. Rationale for the Exchange of Informal Knowledge

The following points answer the question: Why should we want to include informal knowledge in a KEL?

a) So knowledge bases lacking sufficient formalization, e.g. those under development, can be exchanged:

In our research, we have found it very useful to have knowledge enterers 'sketch' knowledge initially and then gradually formalize it over time. This is especially important when using a knowledge based

8 As a matter of theoretical interest, we believe that a knowledge representation should be able to have *no* informal knowledge (unlike most current representations). This means, in particular, a representation with *no* explicit permanent symbols. One can construct a CODE4 knowledge base with no permanent symbols: Internal references are memory pointers, and temporary symbols are generated only for display or storage.

system as a creative tool, as a brainstorming tool, or as a tool to resolve differences among experts. As an improvement over natural language documents, it is essential in our experience to be able to exchange such knowledge bases.

At the current time, most typical knowledge acquisition systems will not tolerate any syntactic freedom, except those that permit unconstrained natural language. We believe it would be desirable for there to be a continuum; CODE4 is designed with this in mind: Very fine-grained elements of knowledge can be either formal or informal, and can be transformed between these states through various mechanisms. Larger knowledge structures, composed of the finer-grained elements, can therefore be partially formal and partially informal.

One should distinguish 'sloppiness' from 'informality'. Stored knowledge can be informal e.g.,

• if only fragments are present, e.g. "you fill in the details"

• if the syntax has not been refined, e.g. an outline of a rule,

• if concepts that are referred-to have not been defined, or

• if the knowledge is merely commentary in nature.

Sloppiness, on the other hand, refers to such undesirable activities as making statements or choosing terminology without careful thinking, or without careful review.

b) Because it is necessary, at some level, to *ground*[6] knowledge in terms of human- or machine-understandable elements:

Even if knowledge is entered very formally from the start, it is still necessary to make some of the representation human-or machine-interpretable, otherwise nothing can be done with the knowledge. At the very least this implies allowing human-understandable symbols (and all known systems allow this). This idea should be extended, e.g. to allow images.

c) Because informal elements allow for heterogeneous knowledge representation systems to coexist:

It is sometime desirable to store and exchange representations of expressions interpretable only by some specific system, e.g. a particular Smalltalk, or a particular KL-ONE derivative (the representations are therefore *informal* with respect to some system that is not based on Smalltalk or is not the particular KL-ONE derivative). Without this, it would be necessary to ensure that there is a complete mapping between the formal semantics of any two systems. When such elements are loaded into a system that can interpret them they will regain their formal interpretability. When loaded into some other system, they will be given no special treatment, i.e. they remain informal.

d) Because informal elements facilitate research into knowledge representation without hindering it with excessive or unnecessary syntax and/or semantics:

This point follows from point c. Research into knowledge representation is still very young. Different researchers have many ideas about what should be included in a KEL. Most researchers do, however, agree that something equivalent to first order logic would be a minimum requirement. Systems would use informal elements to exchange knowledge corresponding to their special features. Different subsets of the systems could evolve to interpret each other's special informal elements. In time, it might be decided to extend the KEL.

The KIF proposal [GENES 91], for example, supposes that a substantial number of Lisp functions should be primitives in the KEL. We would prefer to only include those functions having some degree of universality (e.g. for which there is an obvious analog in CODE4, or Prolog or C++). On the other hand most existing systems would have a very hard time dealing with some of the features of CODE4, e.g. its support for many-to-many relationships between terms and concepts (CODE4 properly distinguishes these two very different notions [LETH 91]). This could be made an informal feature of a first-generation KEL.

Interestingly, several enhancements to CODE4 have been made without changing the syntax of the CKB format at all (for example the storing and loading of images is done using already-existing informal elements). We anticipate adding informal knowledge about visual appearance and arrangement without further changes to CKB format.

4. Representing Informality in a KEL

Before suggesting how informal knowledge might be represented in a KEL, we believe it is important to give some criteria for evaluating the effectiveness of a KEL's schema. Current proposals (e.g. [GENES 91]) do not offer such criteria, so what follows is offered for general consideration.

We propose that a KEL and compliant systems satisfy the following criteria:

a) Conservation of system-specific knowledge by the KEL:

A KEL knowledge base written by a compliant system shall be subsequently readable by the same system. The resultant internal representation of both informal and formal structures shall be functionally equivalent to that prior to the the writing, i.e. nothing shall be lost.

b) Conservation of KEL knowledge by a compliant system:

A compliant system X shall be able to read any KEL knowledge base originated by another system Y, and shall be able to subsequently write out an equivalent KEL knowledge base that is functionally equivalent to that which was read, i.e. X shall preserve any special-purpose informal knowledge originated by Y.

Criterion a) requires preservation of system-specific semantics of knowledge despite the fact that the knowledge is translated to KEL and back. Any semantics not defined in the KEL standard would have to be transmitted by informal structures.

Criterion b) requires that knowledge structures, including informal ones, survive manipulation by a system potentially different from the one that had originally written the knowledge base.

The main conclusion we can draw from the above is that the representation of informal elements must have the following properties:

• Informal elements must be syntactic units that a parser will unambiguously recognize as such.

• Specific systems may choose to tag informal elements with unique flags, (e.g. tags universally understood t mean 'Common Lisp', 'Smalltalk block', 'Compressed image', 'Postscript', 'CODE4 term list').

When processing a KEL file, a system should take an informal element as a whole. Some systems may decide to further process the informal element, others would ignore its contents, preserving it for when a new KEL file is to be written.

In CODE4, we pre-process informal elements when writing CKB format so they can be expressed in ASCII strings that do not include the character that unambiguously terminates the parsing of the informal element. When reading CKB format, some processing is attempted (e.g. to convert compressed images back into displayable images), but if this fails, the informal element is kept as a simple string and is displayed as such. A given non-CODE4 system that could read CKB format might not be able to interpret compressed-image strings, but no knowledge would be lost. Instead, compressed images could at least be viewed (as funny strings) by the user in addition to regular strings.

5. Conclusions

In this paper we propose that serious thought be given to allowing the representation of informality during the development of a knowledge exchange language (KEL). Our work with CKB format in CODE4 has led us to several important conclusions:

a) The KEL should be able to store informal elements as syntactically distinct units, and compliant systems should be able to process these or merely store them for later output.

b) The informal elements should contain knowledge that has no agreed-upon formal semantics, but may be semantically meaningful to humans or specific systems.

c) The core semantics of a KEL should be very small, such that all members of the knowledge representation community can agree about it. The representational needs of systems that go beyond the core semantics can be handled using metaknowledge and/or informal elements.

Acknowledgements

The CODE4 system has been developed by the authors as well as Ken Iisaka, Danny Epstein and David Corbett. Ken Iisaka designed the CKB syntax in the appendix. Support for this research has been provided in part by the Natural Sciences and Engineering Research Council of Canada.

Appendix

The following is the syntax of the CKB file format, by which CODE4 systems exchange knowledge bases. CKB format is very compact, but includes only printable ASCII characters so it can be emailed and processed easily by other software. It is not designed for direct human readability, but can be easily followed if necessary (e.g. for debugging, or developing a new system).

In lieu of symbols (which are not strictly necessary), references between concepts in CKB format are made using integer labels. The integers are sequentially assigned as a knowledge base is written out, and are discarded on input.

CKB format supports informality in four ways, corresponding to the points made in section 2:

i. Symbols are all represented in the string of <Term>. This string can be interpreted as a simple character string, or an arbitrary image. Concepts can have any number of terms, including none.

ii. Order is preserved in collections. At the current time there is no provision for storing general spatial arrangement or other graphic attributes, but it is planned to add this to CODE4 without changing CKB format: Graphic information about a concept would be stored as a term (recognizing the fact that appearance or orientation is a means of conveying information symbolically).

iii. Arbitrary strings can be entered in the <head> of <CleartalkNode>. CODE4 may or may not be able to parse this string to obtain <semantics>. In fact, the parsability may dynamically change as knowledge is altered: The formality thus dynamically changes.

iv. Comments can be associated with any concept.

The internal syntax of strings (which can be compressed images) is not shown. It was not considered necessary to distinguish this functionality in the semantics of the exchange language

Some 30 small knowledge bases have already been produced with CODE4 in our laboratory. An extended version of CODE4 is now also being developed by Boeing [BRAD 91], and they plan to use CKB format. We anticipate they will be able to store their extensions in informal elements, so that their knowledge bases will still be readable by the original system.

CKB format is still under development: We do not propose it as a KEL, we merely hope it stimulates ideas. Although it has syntactic markers for specific types of concepts used in CODE4, these could very easily be removed.

```
<KB File>            ::= <header> <conceptLists>

<header>             ::= ''COMMAND KB'' <cr>
                         <systemVersion> $,
                         <KBversion> <cr>

<systemVersion>      ::= $V <digits> $. <digits>

<KBversion>          ::= <digits>

<conceptLists>       ::= | <conceptList>
                         <conceptLists>

<conceptList>        ::= $( (| <concepts> ) $)

<concepts>           ::= <conceptRecord>
                         | <conceptRecord> $,
                         <concepts>

<conceptRecord>      ::= ( <Type> | <Instance>
                         | <Property>
                         | <Statement> | <Term>
                         | <PrimitiveType>
                         | <PrimitiveProperty>)
                         <terms>
                         <sourceProperties>

<Type>               ::= $Y <subconcepts>
                         <disjunctions>

<subconcepts>        ::= <collection>

<disjunctions>       ::= <collection>

<collection>         ::= $[ ( ε| <elements>) $]

<elements>           ::= <nil> | <digits>
                         | <digits> $ <elements>

<nil>                ::= $@

<Instance>           ::= $I

<Property>           ::= $P <subproperties>
                         <superproperties>
                         <statements>

<subproperties>      ::= <collection>

<superproperties>    ::= <collection>

<statements>         ::= <collection>

<Statement>          ::= $O <subject>
                         <predicate> <value>
                         <status> <modality>
                         <comment>
                         <knowledgeReference>

<subject>            ::= <collection>

<predicate>          ::= <collection>

<value>              ::= <collection>

<status>             ::= <string>

<modality>           ::= <string>

<comment>            ::= <collection>

<knowledgeReference>::= <collection>

<Term>               ::= $T <string>

<MetaConcept>        ::= $M <submetaconcept>

<PrimitiveType>      ::= $y <subconcepts>
                         <primitive>

<primitive>          ::= <string>

<PrimitiveProperty>  ::= $p <subproperties>
                         <superpropertise>
                         <statements>
                         <primitive>

<terms>              ::= <collection>

<sourceProperties>   ::= <collection>

<CleartalkNode>      ::= $C <CleartalkClass>
                         <semantics> <head>
                         <CleartalkInstVars>

<CleartalkClass>     ::= <classKey>
```

```
<semantics>          ::= <collection>
<head>               ::= <string> | <digits>
<CleartalkInstVars> ::= | $[ <CTInstVarList> $]
<CTInstVarList>      ::= <CTInstVar> |
                         <CTInstVar> $,
                         <CTInstVar>
<CTInstVar>          ::= <nil> | <digits> | $(
                         <collection> $)
```

References

[BRACH 91] Brachman, R., McGuiness, D., Patel-Schneider, P. and Alperin Resnick, L., "Living with Classic: When and How to Use a KL-ONE-Like Language", in Sowa, J ed., *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, Morgan Kaufmann, pp 401-456

[BRAD 91] Bradshaw, J.M., Ford, K.M. and Adams-Webber, J., "Knowledge Representation for Knowledge Acquisition: A Three-Schmata approach", *proc. 6th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, October, pp 4-1:4-25.

[GENES 91] Genesereth, Michael R. and Fikes, Richard, "Knowledge Interchange Format, Version 2.2, Reference manual", *Logic-90-4*, Logic Group, C.S. Dept, Stanford Univ., March.

[LENA 90] Lenat, D. and Guha, R. *Building Large Knowledge Based Systems.* Reading, MA, Addison Wesley

[LETH 91] Lethbridge, T.C., "Creative Knowledge Acquisition: An Analysis", *6th Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff*, October

[MUND 91] Mundie, D.A., and Shultis, J.C., *proc. Workshop on Informal Computing*, Santa Cruz, May, Incremental Systems

[SCHMO 91] Schmolze, James A. and Mark, William, "The NIKL expreience", *Comput. Intell.* 6, 48-69

[SKUC 91] Skuce, D., "A Wide Spectrum Knowledge Management System."*To appear in: Knowledge Acquisition* : 49 pp

[SKUC 92] Skuce, D., and Lethbridge, T.C., "A Knowledge Represenation for Interactive Knowledge Management", *in preparation*

[SOWA 84] Sowa, John, *Conceptual Structures: Information Processing in Mind and machine*, Addison-Wesley