# Towards a WorldWide Knowledge Base
# Extended abstract

Erik Sandewall
Department of Computer and Information Science
Linköping University
S-58183 Linköping, Sweden
erisa@ida.liu.se

**Abstract.** The paper proposes that the basic idea of the WWW can be generalized to the domain of data structures or information structures. This means that "packages" of structured information are represented as short text files, stored in web-accessible locations (URL:s), and are loaded as needed by the web-oriented database tool. We suggest that this technique may help to get done what has not been achieved before in the history of A.I. research: a comprehensive, universally usable knowledge base of common knowledge.

## 1 Questions

Why doesn't there exist, after 35 years of research in artificial intelligence, a comprenhensive knowledge base of common-sense knowledge which everyone can use as a starting-point for her projects? Since "symbolic", "representation-oriented", and "knowledge-based" approaches have dominated throughout the history of A.I., why is it that every research group and every project builds its own knowledge base more or less from scratch, to fit its particular application area? If common-sense knowledge is truly universal and can not be cut into small self-contained pieces, why is it that almost every project that professes to use a knowledge-based approach does exactly that?

And, if we agree that the present situation is unreasonable and that a comprehensive universal knowledge base is a necessity for A.I., what are the possibilities of changing the situation by using the new software technologies that have emerged in recent years, in particular, the World-Wide Web technology?

The present paper attempts to give a brief answer to the first set of questions, and a more extensive answer to the second set.

## 2 Why don't we have a comprehensive knowledge base today?

The present absence of one or more widely used knowledge bases is of course due to a combination of several factors. I will therefore only identify a few aspects of the relevant background.

*Disdain for trivial information* bears a great part of the blame. We act as though there were a strict difference between data bases and knowledge bases, where data bases would deal with large volumes of uniformly structured data, so they were trivial from an A.I. point of view. A.I. would deal with Real Knowledge, which is known to have a Very Rich Structure.

The problem with this point of view is that it leads us to only consider Quite Complicated Solutions, in particular, quite complicated ways of structuring information. The view that simple techniques can't be A.I. has two consequences: it gives us structures that are hard to agree upon, because everyone has his own opinion about how to solve the difficult problems. Also, it focusses the attention on the design of how to structure the information and how to build software that operates on it, leaving insufficient room for efforts to actually build the knowledge base.

There are exceptions, of course. What about the CYC project? True, it has attempted to actually build an encyclopaedic knowledge base. However, it has chosen a quite advanced structure, making it difficult for others to embrace it. Also, the decision to make the knowledge base proprietary is a major problem from the point of view of the research community. Where would UNIX be if it had been proprietary from the start?

The KIF effort seems to suffer from a similar problem: it provides a quite expressive language, but it invites dissent about many parts of its technical solution. Efforts to standardize terminological languages may be more promising and we return to them below.

The importance of starting with simple solutions has been demonstrated again and again in the history of software: big and complex products hardly gain acceptance, even if they are so good that they solve all the customer's problems (which of course is unlikely anyway). The best way to win is to begin with something *very simple* that solves *some* of the customer's problems, to gain acceptance for it and to obtain a "customer base", and then to gradually extend the capability of the tool.

In the case of data bases vs knowledge basis, it would help to recognize three levels rather than two. For simplicity I call them "data bases", "information bases", and "knowledge bases", respectively:

– *data bases* deal with uniformly structured data which are strongly typed: the database schema specifies which properties are allowed for which objects, what must be the type of each property, etc.
– *information bases* deal with data which are structured, for example as property-lists (to use the Lisp term), but which are not strongly typed using "declarations" or "schemas". Still, the kinds of information that are represented there are fairly simple, and the data volumes are moderate.
– *knowledge bases* are as we know them from A.I., requiring solutions to the large number of problems for representing e.g. common-sense knowledge.

I submit that knowledge bases are rarely useful unless they are combined with information bases: there is a lot of routine information around which must be present together with the very flexible contents of the knowledge base proper.

The need for the middle level, information bases, has also been observed from time to time in the database community, with proposals for "binary data bases", "object-oriented data bases" (in one sense of the phrase "object-oriented"), and so forth. However, this direction does not seem to ever have established a strong life of its own in that community either.

## 3 The virtue of simplicity

On the background that has just been described, one obtains at once the following strategy for the future:

- Let us define a simple initial format for expressing information (in the above sense of that word). The format should be so simple and straight-forward that acceptance is easy, and implementation of the basic software support is trivial.
- Build up stores of common-sense information using that format. Examples of such information abound: basic geographical information (countries, cities, languages, etc), basic historical and political information, and so on.
- Obtain maximal availability for these information stores. In particular, organize them in such a way that different contributions can easily be combined.
- Identify a few early, useful application areas for the totality of common-sense information that has been built up, and demonstrate these applications widely.
- Once this has been done, add more facilities to the basic initial format. Preferably, the additions should be done in a modular fashion, so that they can selected or deselected somewhat independently.

From the point of view of practical computing, this strategy is elementary and very obvious, but it differs considerably from the common approaches in A.I.

The first items in the list are not difficult. Basic set-theory notation offers a very natural candidate for the notation: sets, sequences, and mappings provide a well-known, versatile, and yet very simple notation. One should also assume a few distinct types for the elements of sets, sequences, etc, corresponding to the atomic data types in Lisp. Thus one needs numbers, strings, and symbols, and then the composite structures (sets, sequences, mappings) that can be constructed from them. – The now-standardized notation for terminological languages is another likely candidate.

The crucial issues are availability and first applications. This is where the World-Wide Web comes in.

## 4 Universal availability

Consider the researcher who has built up a store of common-sense information according to the simple principles that have just been described. How does he distribute it to others?

In older days, until a few years ago, he or she would have written a few papers about it, and offered to distribute the software and the data bases by the file transfer methods of the day, typically by ftp. Distribution would be relatively slow; any prospective user would have to import both programs and data, try it out, and then figure out how to interface with his own software or software provided from other sources.

After the advent of WWW, it can be done quite differently. Consider the difference between using a WWW browser and using ftp: in ftp, you have to specify by separate interactive commands each of the file transfers you wish to make. In the browser, you are working *inside* the information, and you get the browser to load additional modules (additional "pages") as it needs them.

Translate this to the database or information-base context. In the old style of working, one would transmit the whole information base from its originator to its prospective user. Now, it is feasible to have an information-base tool which keeps a certain collection of information "packages" in working memory, and which loads additional packages from the net as it needs them. Each information package should be able to contain a certain amount of information (for example, a property-list for a particular object), but packages must also be able to contain links to other packages *located at other hosts*, just like HTML pages do.

In this way, it becomes possible to emerge an information base by the many contributions of users everywhere on the Internet, just as the world-wide web structure has emerged from a very large number of interlinked, individual contributions.

## 5 First applications

Besides the technical solution, there is also a need for good first applications. Interestingly enough, this aspect is also provided in the context of the world-wide web.

The basic observation is that it is fairly inconvenient to write WWW pages directly in HTML. There are at least two alternative ways of dealing with this problem. One possibility is to build interactive tools which allow the user to drag icons, choose from pull-down menues, etc. The other solution is to maintain a small database or, in our terms, an information base as the primary store of information, and to generate WWW pages from the data base. Both approaches have their pros and cons, but there are a number of specific advantages for the generative approach, just as Emacs/Latex has advantages over WYSIWYG editors. For example, if a particular name, word, or phrase is to be associated with the same home page and the same URL in many different occurrences, then it is convenient to generate automatically the pages where these references occur.

The first applications of such an information base can therefore be to maintain information from which world-wide web pages in HTML are generated.

## 6 Experimental implementation

The approach which has now been described in outline has also been implemented, and has been described in more detail in other papers which are accessible from

`http://vir.liu.se/brs/database/`

The following implementation technique has been used. Each information package is represented as a text file under Unix, using a Lisp-based notation. We have implemented an information-base software tool which reads these text files as it needs them, just like a web browser does, but for structured data. The tool has been implemented in Xlisp, and is presently being used for two purposes: for E-mail management, and for maintenance of the WWW pages of the ECSTER electronic colloquium. ECSTER stands for "European Colloquium on Spatial and Temporal Reasoning", and is run within a special interest group of Compulog. To get a feeling for the generated web pages representing the information structure, please refer to

`http://vir.liu.se/brs/`

and the pages that are referenced from there.

## 7 A programming-language perspective: the eternal Lisp

The programming language Lisp, which was used for this implementation, has been the workhorse of A.I. research ever since the area started. Its use in the approach being described here is not accidental, nor is it based on conservatism or nostalgia. On the contrary, it is motivated by the most characteristic property of Lisp: *the existence of a convention and corresponding software support for representing data structures as text.* This property of Lisp is the fundamental reason for its persistence, in my opinion, and it has not yet been incorporated into widely used languages such as Pascal or C++. It has also not been incorporated into the favored newcomer today, that is, Hot Java.

However, Hot Java is interesting from another point of view in this context. If the key invention in WWW was to allow small modules of marked-up text, which could be loaded as needed into the browser, the corresponding generalization in Hot Java is to allow procedures or other fragments of programs to be treated in the same way: requested as needed, and loaded into the interpreter. The present proposal extends the same principle to a third dimension: not only marked-up text and program fragments, but also packages of "information" (data without the restrictions of strong typing) can be treated in this dynamic way.

## 8 From information base to knowledge base

The last question from the impatient A.I. researcher may then be: fine, this will give us a world-wide information base, but when and how will it develop into the world-wide knowledge base that is alluded to in the title of the paper. The answer is simple: *just wait.*

## 9 References

Please refer to the URL (WWW address) mentioned in section 6 for the full versions of my papers on this topic (including papers that may have appeared after the present article), as well as for references to related work.

This article was processed using the LaTeX macro package with LLNCS style