THE ROLE OF SOFTWARE IN DIGITAL LIBRARIES, AND THE ROLE OF DIGITAL LIBRARIES IN SOFTWARE

M E Lesk

Rapporteurs: Robert Allen

The Role of Software in Digital Libraries, and the Role of Digital Libraries in Software.

Michael Lesk

Bellcore
Morristown, NJ
USA

ABSTRACT

Today we are making substantial progress in building digital libraries[1]. We have vast online files, CD-ROMs of major journals, and over 100 text browsers for sale. Some of the digital library systems are based on Ascii text, and some on images of pages. Both methods can produce material that is more effective at a user's task than is information on paper. Surprisingly, however, we are making less progress on software libraries to support code re-use and permit us to build on each other's work. Despite the fact that computers can understand code, and not English, we can automatically index or classify English, but not code. Fundamentally, code is a very abstract language. Research is needed to improve our ability to recognize what is in code, and sort it effectively.

1. Introduction

In 1994 the United States started the Digital Library Initiative, funded by the National Science Foundations (NSF), National Aeronautics and Space Administration (NASA) and the Advanced Research Projects Agency of the Defense Department (ARPA). This solicited research proposals for digital libraries, and made six awards. Of the six, four plan to use as their collection scientific papers. The University of Illinois will do scientific journals in general; the University of Michigan is focussing on earth and space sciences; Berkeley will do California environmental information; and Stanford's collection starts with computer science literature. The other two are less usual: the University of California Santa Barbara is building a project based on images and maps, and Carnegie-Mellon University will be using television broadcast videotapes.

Digital libraries could, of course, contain other kinds of information. The rejected proposals are not made public by the granting agencies, although some of the universities have discussed their plans. There were, for example, at least two proposals for data libraries, which would have been interesting research. Someone may have proposed a sound library, storing acoustic information instead of printed information.

But a particularly interesting possibility possibility would have been a software library. Large software projects only occasionally arrive on time and on budget; *Scientific*

American for September 1994 estimated that only about 1/4 of large software projects are successful. If we could rapidly find software that performed particular functions, we could expect to increase software re-use and facilitate software development.

Part of the problem is not about code, but about industrial databases. Manufacturers maintain large databases of the parts they use in their products (John Deere, for example, has 750,000 parts in its files) and have similar difficulties retrieving particular items. But our inability to build libraries of code is particularly baffling. Code, unlike CAD drawings or English text, is completely definable by a computer; it really has no other meaning. And yet, we are unable to pick up a piece of code and decide that it should be put with the matrix diagonalization routines, or with the plot drawing routines, or what.

How do text libraries work? There are two general ways. One set of possibilities relies on manual indexing, in systems such as MEDLINE. The set of people who index medical literature has no analogy in the computer field. Don Knuth argued that we should publish programs as literature; not only do we not do so, we don't index them, either. The other, and now more frequent kind of searching, relies on the individual content words of English. This article is about digital libraries because it contains those words.

Computer code, by contrast, does not contain the equivalent of function words. The functionality of code is in its structure, not its particular words. If we considered code as a language, it contains many more function words, very short sentences, many more inter-sentence links, and most important, only proper nouns or pronouns. A proper noun (e.g. *John*) only really has meaning by pointing to a particular person, whereas a common noun (e.g. *apple*) has a meaning independent of the context. Computer variables are more like proper nouns or pronouns; it does not make sense to index documents by words like *John* or *she*, it only makes sense to index by words like *apple*. Since most programs consist of variables, we do not have the information needed to access them.

If we could build access tools, however, they would be very useful. Large software systems now as big as the largest book. Bellcore, for example, maintains 70M lines of code. Since a book usually has 60-70 characters per line, this implies that Bellcore maintains the equivalent of about 5,000 books. Needless to say, if you added up all the ordinary (English) books written and published by Bellcore staff, it would be far fewer than 5,000 titles (at most a few dozen).

2. Digital Libraries

Information browsing may well be the ''killer ap'' for the global information infrastructure. After a long period of building text retrieval systems for specialists, ordinary people are now browsing and searching the Internet. Although they do not know it, they are using hypertext, word associations, and ranked retrieval, among other algorithms. In addition, large numbers of ordinary library users, who had often refused to read off screens when those screens displayed images from microfilm, are now reading off screens that contain images from CD-ROM. Readers are browsing rather than just searching for specific items, and they are not dealing just with text, but with pictures and sounds; in fact it is the advent of pictures and sound that have made the Web so attractive to so many.

The explosion of information on the Web is widely known. There are something like 5 million web pages found by Lycos, and CMU estimates that there might be 20 million total web pages. Netnews is distributing, every day, 450 MBytes in 140,000 articles. This is up from 50 MB a day in 1993 and 1 MB/day in 1989. If we assume that a year's worth of book publishing is about 50 GB, this means that Netnews is distributing more bytes than the entire US printed book industry.

Libraries, and other systems for dealing with scholarly information, are changing as well. Some libraries, especially corporate libraries in industries such as pharmaceuticals, already spend more money buying bits than buying paper. The data industry as a whole, including online financial information, is now comparable to the nonfiction book industry. Price increases, space shortages, budget constraints on universities, and other problems are steadily shrinking the amount of printed literature a typical academic library can buy and keep. The current system is near collapse, but at the same time online catalogs have now taken over, interlibrary loan largely moves by fax rather than by mail, and CD-ROM sales doubled each year in the early 1990s.

To show just how far the economics of digital libraries have come, consider just the costs of building libraries against the cost of scanning books. The Cornell CLASS project reported costs of about $30 for scanning books[2], and these were fragile-paper books requiring that each sheet be placed by hand on a flatbed scanner. Substantially cheaper scanning would be possible for books with paper strong enough to fed through a mechanical stacker. Even assuming a $30 cost, however, and adding the less than $10 of disk space required to store the resulting books, remember that this is only needed once, no matter how many libraries wish to access the books. By comparison, a new bookstack at Cornell cost about $20/book and at Berkeley, about $30/book (the extra cost of construction to stand up to earthquakes). The new library building at UCSF is about $60/book, the new BL about $75, and the new French national library about $100/book. These latter libraries are more than just bookstacks, however; they include reading rooms, offices and so on. If the books are online, however, you might not need the reading rooms either. In any case, today, if four universities which were all planning to build a new stack could instead identify a few hundred thousand out-of-copyright books they all held, scan them, and remove the originals to make space, they would be financially better off. The Mellon foundation has funded just such a project, project JSTOR, in which the entire run of ten important economics and history journals will be scanned (with permission granted by the publishers to cover the more recent issues which are still in copyright). Libraries will then see if they can successfully substitute access to these images for their copies of these widely held journals.

Why don't libraries already do this? Part of the reason is difficulty with obtaining copyright permission to scan the more recent and valuable material (which even if out of print, remains in copyright for 75 years under the present law). Part of the reason is lack of viewing stations in the libraries and among the users (and networks to connect them). But part of the reason is fear that the users can not, or will not, use the images in place of the paper. Some years ago, we began an experiment to measure the usability of online files.

3. The CORE Project – Testing Online Primary Journals

The CORE project[3] is an experiment in online access to chemical journals, and it is a joint project of Cornell University, the American Chemical Society (ACS), Bellcore, Chemical Abstracts Service (CAS), and OCLC (the Online Computer Library Center). We wanted to compare how effectively people could use paper and electronic journals, and we also wished to compare the effectiveness of image and Ascii representations of journals. Some digital library projects have been using images of pages; these include, for example, the Elsevier TULIP experiments[4], the commercially available IEEE journals on CDROM from UMI, the Adonis system[5], and the AT&T/UCSF Red Sage project[6]. Other projects use Ascii text as their base: this includes the entire Web but also such products as the Chadwyck-Healey English Poetry Database and the STN system (Science and Technology Network) and the full text sources available through Dialog, Nexis, and other vendors. Full text for primary journals has been delayed, unfortunately, by difficulty of including graphical material. Some systems (e.g. OCLC's online journals such as the Online Journal of Current Clinical Trials) do support graphics, but many of the commercial offerings do not. On the other hand, the CD-ROM systems are normally only available in libraries, not at the user desktop, since either legal or technical reasons restrict the ability of the libraries to distribute the information over a local network.

Our project has one of the few joint databases. We converted some 300,000 pages of ACS journals, both scanning all the pages and also providing SGML derived from the ACS database (ultimately produced as part of the typesetting operation). The ACS journals do contain visual information, about 1/4 of the total journal (measuring in square inches) being graphics. Virtually all the graphics are line drawings, typically chemical structures or spectrograms or drawings of experimental apparatus. We obtain these illustrations by analyzing the scanned images, and finding the material that does not have the regular line structure of text. These images are then combined with the Ascii derived from the database to produce SGML, according to a DTD that is extended from the AAP EMS definition, and then placed into a large database. An overview of the data flow is shown in Figure 1. ACS provides primary journal pagess; CAS provides indexing and magnetic tape of journal text. The pages are scanned, graphics picked out, and the resulting graphics combined with the text content to make a file of text in Ascii plus images. The users express queries against the text file and then see either the image or Ascii version, depending on the interface.

We provided interfaces using both images and Ascii. The simplest interface was the image browsing interface, called Pixlook. Figure 2 is an example of Pixlook, in which the user is browsing through a set of journals. A list of 20 ACS journals is presented, and the user chooses one, and then from other windows selects a year and issue. The table of contents of that issue, showing the authors and titles of each article, is in the bottom left window. Clicking on a line in the table of contents causes the first page of that article to be displayed. Although the scanning is at 300 dpi, one bit per pixel, the initial display is at 100 dpi so that it fits on the screen. It is enhanced with grey-scale anti-aliasing, so that it can be read although not comfortably. The user has buttons to zoom in, so that the page is displayed at full resolution, in which it is easy to read but only a small part of the page fits on the screen. Other buttons move to the next page, or request a printed copy of the article. The actual display of the article, in this interface, is the scanned image of the page.

Pixlook also permits the user to search for words, using a simple Boolean search feature (which includes fielded searching). Because the display is a scanned image, however, there is no highlighting of hits. Fortunately, most of the articles are only a few pages long. The search display is similar to the browsing display: the user enters the search terms in a small window, and then a list of matching articles showing the authors and titles pops up.

Figure 3 shows the SuperBook interface. This is an example of an Ascii interface. SuperBook was designed at Bellcore with a goal of seeing that users could not easily get lost. It views the document collection as a linear string, with a hierarchical organization patterned after the kind of textbook that has section numbers like 2.3.2 or such. The display screen is divided into a left half, which shows the table of contents of the collection, and a right half, which shows a page of text. The table of contents contains the hierarchy, displayed in a fisheye viewer. Each line in the table of contents has the section number and the heading. The fisheye view means that when one zooms in on part of the table of contents to display more detail, the surrounding parts remain in less detail, so that one can keep a large range of the overall table of contents visible. This allows the users to maintain a view of where they are, and avoids the 'lost in hyperspace' problem.

Clicking on any line in the table of contents jumps to the corresponding point in the text, moving the text page to match (and similarly, scrolling forward in the text will adjust the selected line in the table of contents). The text is not an image, but is shown from the SGML (with appropriate fonts, subscripting, etc). Items such as figures, tables, footnotes and equations are indicated by icons in the right margin, and the user clicks on them to bring up an enlarged image. There is one figure indicated on this page, shown as a thumbnail icon.

Searching is unusual in SuperBook. The user can type a word or a short list of words, and the search looks for instances of all of the search terms in one paragraph. There is no way to specify Boolean operators. Since SuperBook has no sense of the collection being made up of articles (the metaphor is that the document collection is a string) there is no hit list in the usual way; instead, the number of instances of the search terms in each part of the table of contents is shown to the left of the headings. This shows the user the distribution of the search term around the collection, and helps decide where to expand the table of contents or read some parts of the collection. Search hits can be highlighted, as shown in this search for "buckyball."

Scepter, shown in Figure 4, was designed by OCLC. It provides a choice of views of the document: it can display either scanned or resynthesized pages; in general it is easier to read the resynthesized page. It provides a Boolean search system; in fact Pixlook performs its searches by calling on the Scepter back end. Scepter helps the user form the search by providing check-off menus for things like the date range or fields to be searched. The result of a search is a hitlist of article citations that match, and then the user can click on the articles to be viewed.

Each article is displayed with a list of the sections of the article: title, author, text, references, tables, figures and so on. The user clicks on this list to bring up the corresponding part of the article. The list of figures is shown as thumbnails, and is the most popular part for most readers. The article text is resynthesized with the extended ACS character set and formatted from the SGML tags.

Dennis Egan and co-workers did an experiment to see how effectively people could use the electronic journals compared with paper journals[7]. Thirty-six Cornell chemistry students were divided into three groups, one using paper journals, one using the Pixlook image interface, and one using the SuperBook ascii interface. Two chemistry professors designed five tasks, chosen to reflect the range of things that chemists need from a library. They also created many instances of these tasks, and each of the students spent six hours doing some of these interfaces in their assigned mode. We measured both the time and accuracy on each task. The experiment used a set of twelve monthly issues of the Journal of the American Chemical Society, containing 1068 articles and 4000 pages.

Of the five tasks, some required searching for information, and some just required reading articles. For example, the simplest task was to find a specific fact in a known article, with a complete cite given. The subjects simply had to go to that article and read it until they found the particular fact. All the students could do this, fairly accurately (about 75% overall scores). We found that it took about five minutes on average for the students to do the reading, regardless of whether they were reading on paper, from the images, or from SuperBook (there was a delay in finding the articles in SuperBook caused by the mismatch between the article and linear string metaphor for the collection). Reading also took about the same time in another browsing task, in which the students were given one issue and asked to skim through it looking for articles on eight topics. On that task, although overall performance was comparable among the three modes of reading, the details of the results showed an interesting difference. The students with the articles on paper made few precision failures (most of what they said they found was actually there) but quite a lot of recall failures (they missed many topics that did appear). The students with the computer access, particular the Pixlook system, made fewer recall failures but more precision failures; that is, they generally found a great deal more, whether right or not.

By contrast, there was an enormous difference in performance on tasks that required searching. The students with the articles on paper, although they had *Chemical Abstracts* on paper to go along with them, had great difficulty searching. They often gave up and were unable to complete the tasks. This reflected partly the difficulty of doing any kind of combined searching (either ANDs or ORs) by hand in a printed index, and partly the difficulties the students had with the controlled vocabulary of *CA*, with which they seemed to have little familiarity. They did have half an hour of training with *CA* (matching the training period they were given for the computer system), but it was not enough.

To oversimplify the results, we found that users of a digital library can read as easily as those using paper, and can search much more effectively. We also found no resentment about the use of the computer systems (in fact one student actually wanted to come back after the experimental session to use the computers for some class project he had to do). This confirms the obvious fact that many people including both ordinary readers and professional librarians like online information and use it regularly. A large industry is developing in electronic books and in encyclopedias, for example, the electronic versions are pushing the paper versions off the market.

5. Software libraries.

As shown above, we know how to build electronic libraries of textual documents and use them successfully. By contrast, we seem unable to make successful libraries of software to be retrieved by content. We are not able to re-use software effectively partly because we can not find the earlier versions. What is strange is that one would think code would be easier to process than text. After all, computer understanding of English, although a dream for many years, has not been achieved, while understanding code is inherent in its nature. For code we can give a completely accurate grammar, a formal definition of semantics, and answer any specific question about its meaning. None of this is possible for English.

Similar problems exist with other kinds of non-textual libraries. For example, it has been difficult to make really general purpose image library systems. There is successful, intelligent image-handling software, but it tends to be specialized: the software for handling CAD drawings, human faces, maps, and so on is not interchangeable. Most libraries traditionally indexed photographs or drawings by writing descriptions of them and then using the text for searching and retrieval. Recently, such projects as the IBM QBIC system[8] have made it possible to imagine that we will have general image retrieval.

There have been some interesting tries at software libraries. One well-known example is Netlib[9[by Eric Grosse, which delivers numerical analysis software by electronic mail. This software library is a good analogy to a manually indexed journal system. Programs are submitted to the library and verified by an editor. The editors are experts in their fields and see to it that junk is kept out of the library. For each program, a short written description is used as an index. Library users retrieve the index and then send requests for the particular items they want; the actual retrieval is entirely automatic. Originally, it operated by email; it now also exists on the net. In addition to the keyword search of the index, there is also now a classification of the numerical routines into a traditional hierarchical structure.

Netlib is very successful (it now has several mirror sites around the world). However, it is an analogy to traditional paper libraries, with manual indexing, and not to automatic subject classification. It operates with volunteer labor, and in other areas of computer science volunteeers have not appeared to make an analogy to Netlib. We still need a way to put arbitrary programs into a library.

A very interesting technology is the dotplot code examiner of Ken Church[10] derived from a technique for looking at biological databases such as DNA sequences. Consider a large matrix in which each row represents one line of code in a program, and each column also represents a line of code. Whenever the row and column contain identical lines of code, the matrix element is 1; if the code is different, the matrix element is 0. Thus the matrix is Boolean, and it is represented by a pixel array with a dark dot for 1 and a blank pixel for 0. Clearly the diagonal line, in which the row and column indices are the same, is always a solid black line. What about the rest of the matrix? Well, some lines of code will accidentally be the same, producing some scattered black dots. But suppose an entire section of code is copied from one place in the program to another. Then there will be another diagonal line, showing a sequence of black dots as matching lines step 1 unit each in both the x and y coordinates.

These short sections of diagonal lines are quickly recognized by eye, and this provides a way to find repeated code easily. This is better than attempts to find such repeated code mechanically, since the line will be recognized even if a few dots are missing (i.e. there have been some changes in the code). Substantial changes can still be expected to produce recognizable links; for example, if a block of code is copied, and then a new line is inserted between every single pair of lines in the copied block, the result will be a diagonal line of dots at a different slope.

There are some difficulties using this technique on large programs. The typical screen can not display a pixel matrix above 1000x1000 and Church wished to use this on millions of lines of code. He developed techniques for abbreviating the matrix (for example, since most of the matrix is white, one can condense it by replacing a square block with a single pixel which is black if any element in it is black). In addition, he was able to use color as a way of helping reduce the size of the image. See the cited paper for details.

This provided an ingenious way to find repeated code sequences in a large program, particularly repeated sequences with slight changes. Simply repeated code sections may be wasteful, but not necessarily dangerous. Repeated sequences with slight changes may mean that code was copied; then a bug was detected and fixed, but that the copy still contains the bug. Thus, the technique has been important to improve the code in the 5ESS telephone switch and other very large programs. An example is shown in Figure 5, which is taken from Church's paper. Note the repeated code sequences indicated by the short downward diagonal line segments.

Although the Church dotplot technique applies to all programs and thus has the generality of our text retrieval systems, it is not a searching system for programs to achieve some purpose. Attempts have been made to build such systems. Leon Shklar at Bellcore built a system called XReUse which used the Latent Semantic Indexing technique to search code, based on the code content, comments, titles, and anything else around[11]. Latent semantic indexing was developed for text files[12]. It took a word-document matrix, and performed a factor analysis to show which words frequently occur with other words. Searching could then be done on the factors, so that a search for a word like ''storage'' might be automatically searching for words like ''disk''' or ''tape'' as well. The same technique can be applied to programs, in which different features of the programs are used to supply the factors. This was done, but the results depend heavily upon comments and titles, effectively returning us to the world of text retrieval.

We desperately need some more techniques for program organization and retrieval. For example, we are having some success at Bellcore with a method called ''cliche recognition'' by its developer, Chuck St. Charles. He looks through old code for patterns which can be recognized as the equivalent of inline macros or subroutines. But this again depends on manually written patterns to start with. More automatic methods would be extremely valuable.

6. Conclusions.

We have techniques that will let us take text on any subject and build a useful retrieval system. We do not have such techniques for software. If you ask, why can't we just pretend software is a language, it has very strange properties viewed as a language. If we imagine that each statement is a sentence, and that operators are prepositions, and most variables are pronouns (since they do not keep there reference point from one occurrence to another), then programming languages have very short sentences, almost all pronouns, and only a few ordinary nouns which have a meaning independent of where they appear. Thus, the standard retrieval techniques don't work on them. We are developing some new methods, but we're not very far along.

As I said at the beginning, it is surprising that we can't do better despite the formal definition of all code. If we look at some function like pretty-printing, for example, we realize that it is easier to do that for code than it is to take untagged English and decide which words should be treated as section headings and put in boldface or which words should be italicized. Yet for the more complex job of retrieval, we can't do that for code.

Part of the problem seems to be mathematical uncertainty in defining what code means. Since we can not even examine a program and decide whether it will stop, we can hardly say what it will do. Theoretically, we can not expect to have a routine which will take a program and pronounce formally and with certainty "this is a sort routine." We might get a heuristic that will do it, and we should certainly look for some, but we're not going to get a provably correct way to do that.

This suggests a sad message for some artificial intelligence research. For decades, it has been suggested that once we have sentence parsing and discourse analysis and a few other language handling techniques, we will have a way to do fully automatic question answering and other applications. The fact that we have the equivalent techniques for programs without being able to do these steps makes one think that success at the more advanced kinds of language processing may be even further away than it now seems.

References

[1] Fox, E. Special issue on digital libraries. *Communications of the ACM*, Association for Computing Machinery, New York (April 1995).

[2] Kenney, Anne and Lynne Personius. *Joint Study in Digital Preservation,* Commission on Preservation and Access, Washington, DC (1992). ISBN 1-887334-17-3.

[3] Lesk, Michael, "The CORE electronic chemistry library," *Proc. 14th ACM SIGIR Conference*, pp 93-112, Chicago, Illinois (October 13-16, 1991).

[4] McKnight, Cliff. "Electronic journals–past, present ... and future?" *ASLIB Proc.*, vol 45, p 7-10 (1993).

[5] Stern, Barrie T. "ADONIS-a vision of the future," *Interlending and Document Supply*, ed. G. P. Cornish and A. Gallico, pp 23-33, British Library (1990).

[6] Hoffman, M.; O'Gorman, L.; Story, G.; Arnold, J.; and Macdonald, N. "The RightPages Service: an image-based electronic library," *J. Amer. Soc. for Inf. Science*, vol. 44, pp. 446-452 (1993).

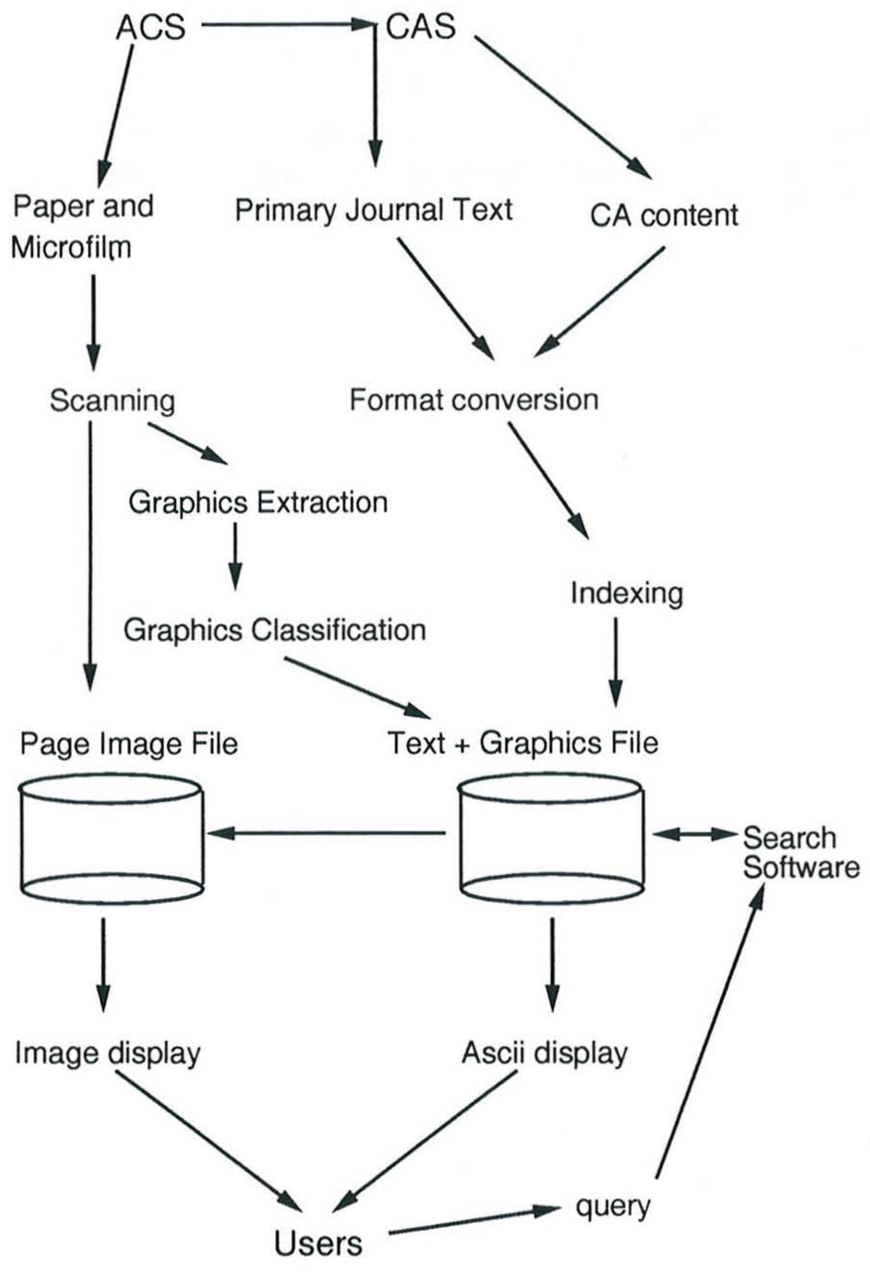
[7] Egan, D.; Lesk, M.; Ketchum, R. D.; Remde, J.; Littman, M. and Landauer, T.; "Hypertext for the electronic library? CORE sample results," *Proc. Hypertext '91,* ACM, San Antonio, Texas (Dec. 1991).

[8] Niblack, W.; Barber, R.; Equitz, W.; Flickner, M.; Glasman, E.; Petkovic, D.; Yanker, P.; Faloutsos, C.; and Taubin, G. ''The QBIC project: querying images by content using color, texture, and shape,'' *Proceedings of the SPIE*, vol 1908, pp. 173-87 (Feb. 1993)

[10] Church, K. W. and Helfman, J. I. ''Dotplot: A program for exploring self-similarity in millions of lines of text and code,'' *Journal of Computational and Graphical Statistics,* vol. 2, pp. 153-174 (1993).

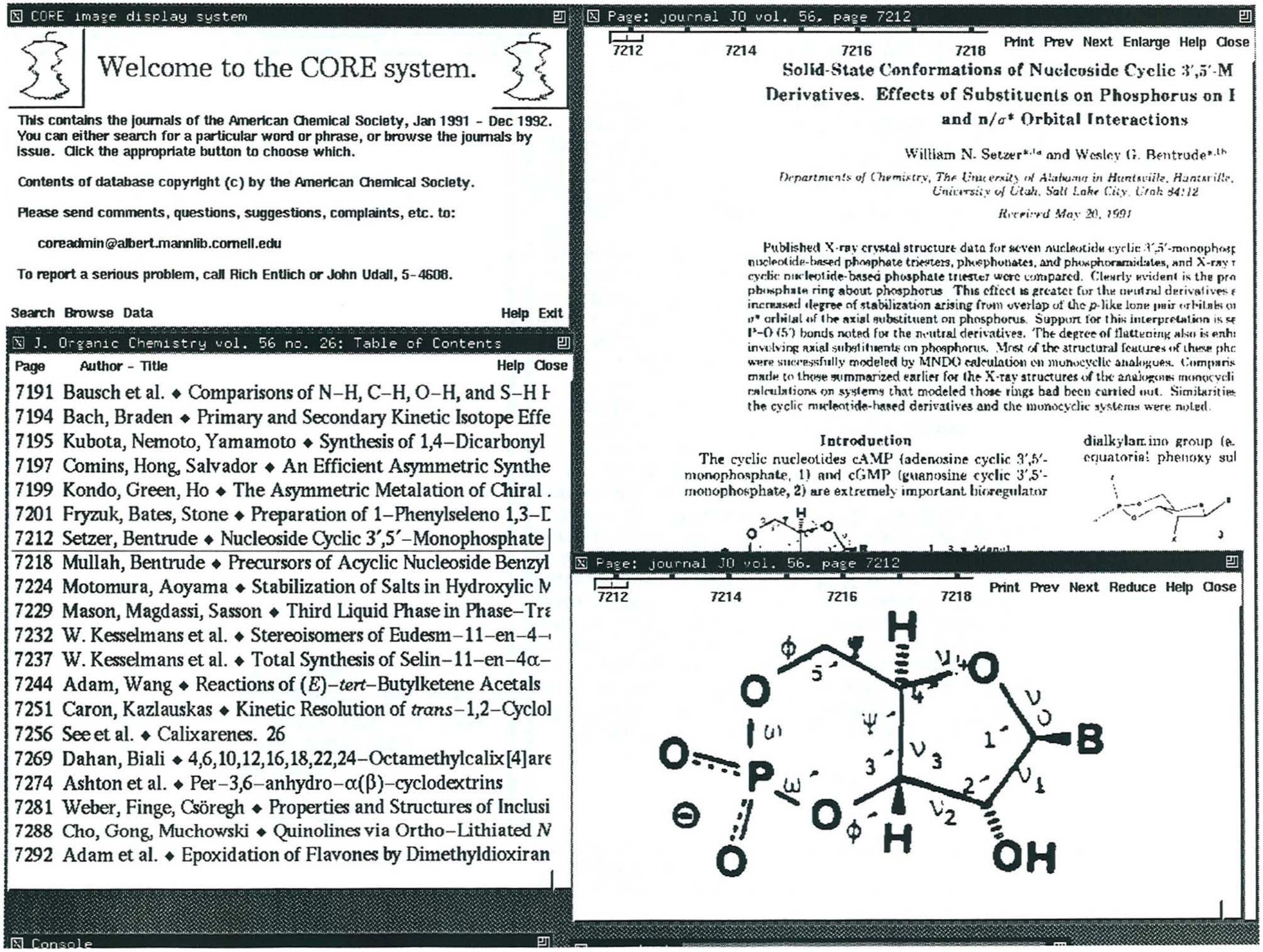
[11] Dumais, S. T. ''Latent Semantic Indexing (LSI) and TREC-2,'' *TREC Text REtrieval Conference (NIST-SP 500-215),* pp.105-15 (1994).

[12] Shklar, L.; Shah, K.; and Basu, C. ''Putting legacy data on the Web: a repository definition language,'' *Computer Networks and ISDN Systems, vol.27, no.6,pp.939-51 (April 1995).*



Data flow in CORE project

Figure 1



Welcome to the CORE system.

This contains the journals of the American Chemical Society, Jan 1991 - Dec 1992. You can either search for a particular word or phrase, or browse the journals by issue. Click the appropriate button to choose which.

Contents of database copyright (c) by the American Chemical Society.

Please send comments, questions, suggestions, complaints, etc. to:

coreadmin@albert.mannlib.cornell.edu

To report a serious problem, call Rich Entlich or John Udall, 5-4608.

Page	Author - Title
7191	Bausch et al. ◆ Comparisons of N–H, C–H, O–H, and S–H I
7194	Bach, Braden ◆ Primary and Secondary Kinetic Isotope Effe
7195	Kubota, Nemoto, Yamamoto ◆ Synthesis of 1,4–Dicarbonyl
7197	Comins, Hong, Salvador ◆ An Efficient Asymmetric Synthe
7199	Kondo, Green, Ho ◆ The Asymmetric Metalation of Chiral .
7201	Fryzuk, Bates, Stone ◆ Preparation of 1–Phenylseleno 1,3–I
7212	Setzer, Bentrude ◆ Nucleoside Cyclic 3′,5′–Monophosphate
7218	Mullah, Bentrude ◆ Precursors of Acyclic Nucleoside Benzyl
7224	Motomura, Aoyama ◆ Stabilization of Salts in Hydroxylic M
7229	Mason, Magdassi, Sasson ◆ Third Liquid Phase in Phase–Tr
7232	W. Kesselmans et al. ◆ Stereoisomers of Eudesm–11–en–4–
7237	W. Kesselmans et al. ◆ Total Synthesis of Selin–11–en–4α–
7244	Adam, Wang ◆ Reactions of (*E*)–*tert*–Butylketene Acetals
7251	Caron, Kazlauskas ◆ Kinetic Resolution of *trans*–1,2–Cyclol
7256	See et al. ◆ Calixarenes. 26
7269	Dahan, Biali ◆ 4,6,10,12,16,18,22,24–Octamethylcalix[4]are
7274	Ashton et al. ◆ Per–3,6–anhydro–α(β)–cyclodextrins
7281	Weber, Finge, Csöregh ◆ Properties and Structures of Inclusi
7288	Cho, Gong, Muchowski ◆ Quinolines via Ortho–Lithiated N
7292	Adam et al. ◆ Epoxidation of Flavones by Dimethyldioxiran

Console
Page: journal JO vol. 56, page 7212
7212
7214
7216
7218
Print Prev Next Enlarge Help Close

Solid-State Conformations of Nucleoside Cyclic 3′,5′-M
Derivatives. Effects of Substituents on Phosphorus on I
and n/σ* Orbital Interactions

William N. Setzer*,[1a] and Wesley G. Bentrude*,[1b]

Departments of Chemistry, The University of Alabama in Huntsville, Huntsville, University of Utah, Salt Lake City, Utah 84112

Received May 20, 1991

Published X-ray crystal structure data for seven nucleotide cyclic 3′,5′-monophosp nucleotide-based phosphate triesters, phosphonates, and phosphoramidates, and X-ray r cyclic nucleotide-based phosphate triester were compared. Clearly evident is the pro phosphate ring about phosphorus. This effect is greater for the neutral derivatives a increased degree of stabilization arising from overlap of the *p*-like lone pair orbitals o σ* orbital of the axial substituent on phosphorus. Support for this interpretation is se P–O (5′) bonds noted for the neutral derivatives. The degree of flattening also is enh involving axial substituents on phosphorus. Most of the structural features of these ph were successfully modeled by MNDO calculation on monocyclic analogues. Comparis made to those summarized earlier for the X-ray structures of the analogous monocycli calculations on systems that modeled those rings had been carried out. Similarities the cyclic nucleotide-based derivatives and the monocyclic systems were noted.

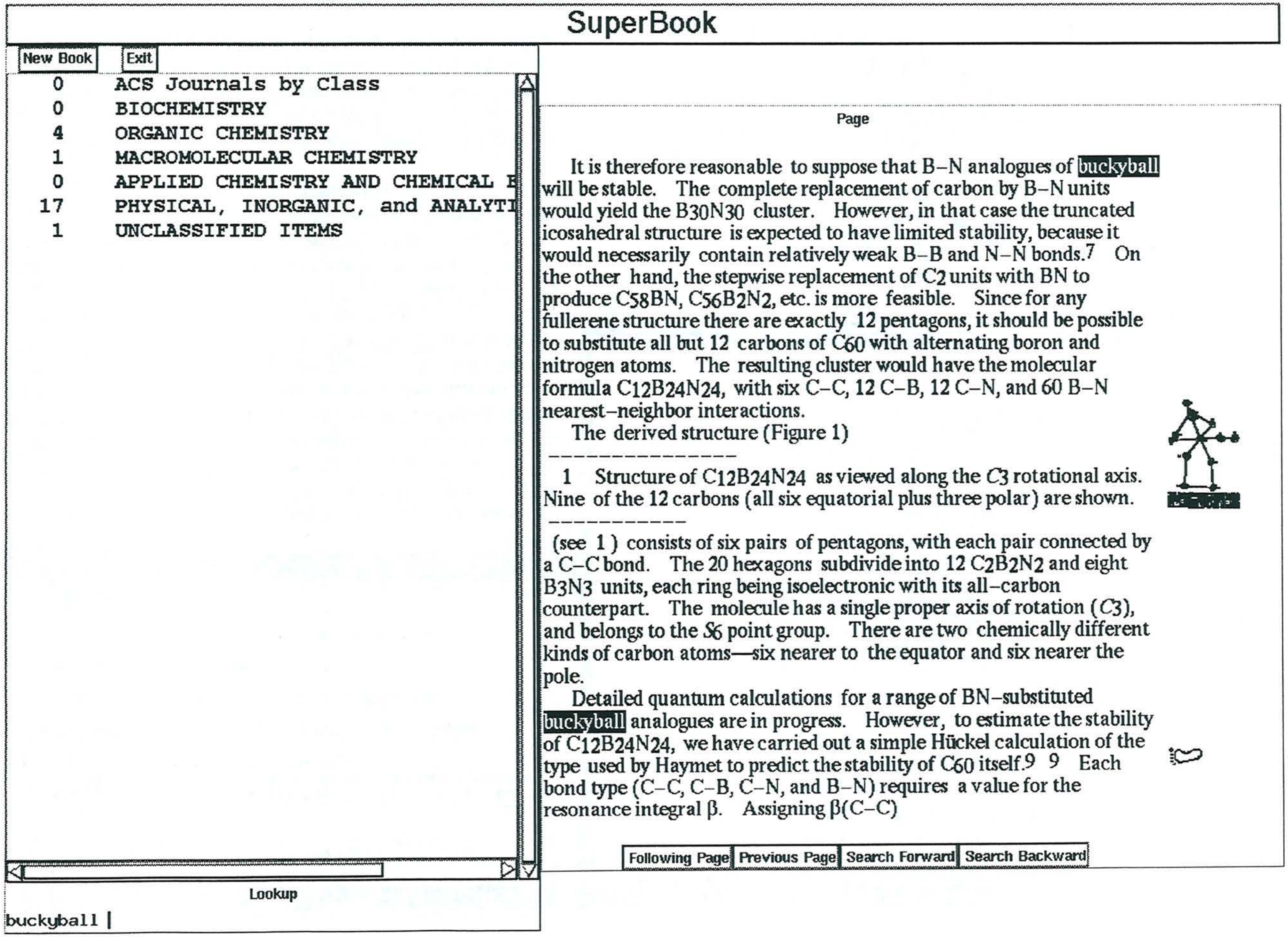
Introduction

The cyclic nucleotides cAMP (adenosine cyclic 3′,5′-monophosphate, 1) and cGMP (guanosine cyclic 3′,5′-monophosphate, 2) are extremely important bioregulator

dialkylamino group (e. equatorial phenoxy sul

Page: journal JO vol. 56, page 7212
7212
7214
7216
7218
Print Prev Next Reduce Help Close
H
O
O
O
P
O
O
H
OH
B
ν₀
ν₁
ν₂
ν₃
ν₄
ψ
ω
φ
1
2
3
4
5

Figure 2. Image display of browsing chemical journals.



```
 0    ACS Journals by Class
 0    BIOCHEMISTRY
 4    ORGANIC CHEMISTRY
 1    MACROMOLECULAR CHEMISTRY
 0    APPLIED CHEMISTRY AND CHEMICAL E
17    PHYSICAL, INORGANIC, and ANALYTI
 1    UNCLASSIFIED ITEMS
```

It is therefore reasonable to suppose that B–N analogues of buckyball will be stable. The complete replacement of carbon by B–N units would yield the $B_{30}N_{30}$ cluster. However, in that case the truncated icosahedral structure is expected to have limited stability, because it would necessarily contain relatively weak B–B and N–N bonds.7 On the other hand, the stepwise replacement of C_2 units with BN to produce $C_{58}BN$, $C_{56}B_2N_2$, etc. is more feasible. Since for any fullerene structure there are exactly 12 pentagons, it should be possible to substitute all but 12 carbons of C_{60} with alternating boron and nitrogen atoms. The resulting cluster would have the molecular formula $C_{12}B_{24}N_{24}$, with six C–C, 12 C–B, 12 C–N, and 60 B–N nearest–neighbor interactions.

The derived structure (Figure 1)

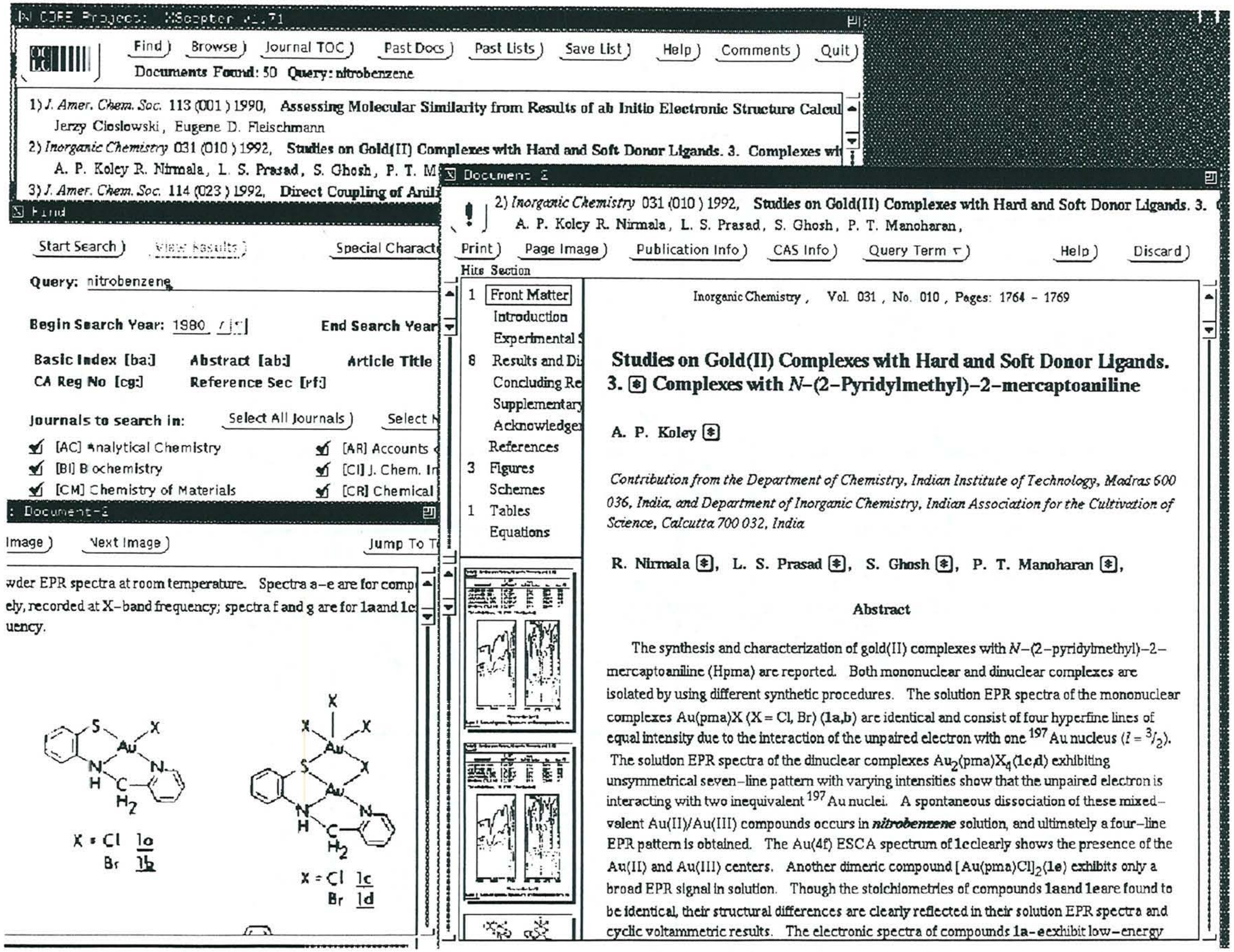
----

1 Structure of $C_{12}B_{24}N_{24}$ as viewed along the C_3 rotational axis. Nine of the 12 carbons (all six equatorial plus three polar) are shown.

(see 1) consists of six pairs of pentagons, with each pair connected by a C–C bond. The 20 hexagons subdivide into 12 $C_2B_2N_2$ and eight B_3N_3 units, each ring being isoelectronic with its all–carbon counterpart. The molecule has a single proper axis of rotation (C_3), and belongs to the S_6 point group. There are two chemically different kinds of carbon atoms—six nearer to the equator and six nearer the pole.

Detailed quantum calculations for a range of BN–substituted buckyball analogues are in progress. However, to estimate the stability of $C_{12}B_{24}N_{24}$, we have carried out a simple Hückel calculation of the type used by Haymet to predict the stability of C_{60} itself.9 9 Each bond type (C–C, C–B, C–N, and B–N) requires a value for the resonance integral β. Assigning β(C–C)

Figure 3. SuperBook screen with contents and text



Find) Browse) Journal TOC) Past Docs) Past Lists) Save List) Help) Comments) Quit)

Documents Found: 50 Query: nitrobenzene

1) *J. Amer. Chem. Soc.* 113 (001) 1990, **Assessing Molecular Similarity from Results of ab Initio Electronic Structure Calcul**
Jerzy Cioslowski, Eugene D. Fleischmann
2) *Inorganic Chemistry* 031 (010) 1992, **Studies on Gold(II) Complexes with Hard and Soft Donor Ligands. 3. Complexes wi**
A. P. Koley R. Nirmala, L. S. Prasad, S. Ghosh, P. T. M
3) *J. Amer. Chem. Soc.* 114 (023) 1992, **Direct Coupling of Anili**

Start Search) View Results) Special Characte

Query: nitrobenzene

Begin Search Year: 1980 End Search Year

Basic Index [ba:] Abstract [ab:] Article Title
CA Reg No [cg:] Reference Sec [rf:]

Journals to search in: Select All Journals) Select N

✓ [AC] Analytical Chemistry ✓ [AR] Accounts
✓ [BI] Biochemistry ✓ [CI] J. Chem. In
✓ [CM] Chemistry of Materials ✓ [CR] Chemical

Image) Next Image) Jump To T

wder EPR spectra at room temperature. Spectra a–e are for comp ely, recorded at X-band frequency; spectra f and g are for 1a and 1c uency.

$X = Cl$ 1a
Br 1b

$X = Cl$ 1c
Br 1d

2) *Inorganic Chemistry* 031 (010) 1992, **Studies on Gold(II) Complexes with Hard and Soft Donor Ligands. 3.**
A. P. Koley R. Nirmala, L. S. Prasad, S. Ghosh, P. T. Manoharan,

Print) Page Image) Publication Info) CAS Info) Query Term) Help) Discard)

Hits Section

1	Front Matter
	Introduction
	Experimental
8	Results and Di
	Concluding Re
	Supplementary
	Acknowledge
	References
3	Figures
	Schemes
1	Tables
	Equations

Inorganic Chemistry, Vol. 031, No. 010, Pages: 1764 - 1769

Studies on Gold(II) Complexes with Hard and Soft Donor Ligands. 3. Complexes with *N*–(2–Pyridylmethyl)–2–mercaptoaniline

A. P. Koley

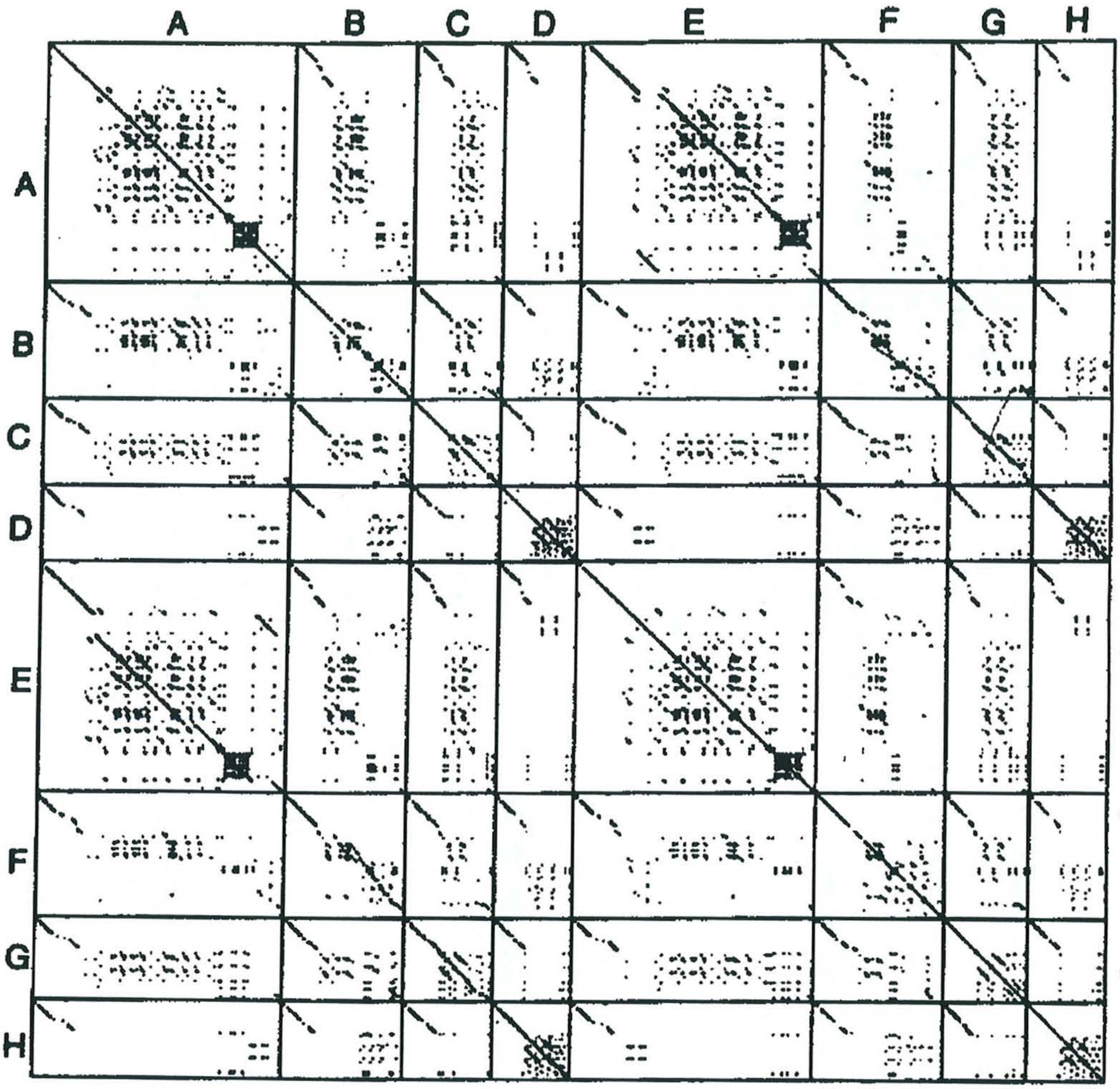
Contribution from the Department of Chemistry, Indian Institute of Technology, Madras 600 036, India, and Department of Inorganic Chemistry, Indian Association for the Cultivation of Science, Calcutta 700 032, India

R. Nirmala, L. S. Prasad, S. Ghosh, P. T. Manoharan,

Abstract

The synthesis and characterization of gold(II) complexes with *N*–(2–pyridylmethyl)–2–mercaptoaniline (Hpma) are reported. Both mononuclear and dinuclear complexes are isolated by using different synthetic procedures. The solution EPR spectra of the mononuclear complexes Au(pma)X (X = Cl, Br) (**1a,b**) are identical and consist of four hyperfine lines of equal intensity due to the interaction of the unpaired electron with one ^{197}Au nucleus ($I = ^3/_2$). The solution EPR spectra of the dinuclear complexes $Au_2(pma)X_4$ (**1c,d**) exhibiting unsymmetrical seven–line pattern with varying intensities show that the unpaired electron is interacting with two inequivalent ^{197}Au nuclei. A spontaneous dissociation of these mixed–valent Au(II)/Au(III) compounds occurs in ***nitrobenzene*** solution, and ultimately a four–line EPR pattern is obtained. The Au(4f) ESCA spectrum of **1c** clearly shows the presence of the Au(II) and Au(III) centers. Another dimeric compound $[Au(pma)Cl]_2$ (**1e**) exhibits only a broad EPR signal in solution. Though the stoichiometries of compounds **1a** and **1e** are found to be identical, their structural differences are clearly reflected in their solution EPR spectra and cyclic voltammetric results. The electronic spectra of compounds **1a–e** exhibit low–energy

Figure 4. Scepter screen display



Ken Church code dotplot

Figure 5

DISCUSSION

Rapporteur: Robert S Allen

Mr Simonyi referred to Dr Lesk's comment that only half of the sale price went towards making the product. He suggested that the retailers also get half of the sale price, and he asked for clarification.

Dr Lesk replied that this wasn't the case in supermarkets.

Mr Simonyi claimed that for, say, furniture, there was such a mark up by the supermarket.

Dr Lesk agreed that this was true for furniture, and also jewellery. He claimed that the big fuss on the net was because some vendors were expecting to live on ten percent of the purchase price.

Professor Katzenelson asked for clarification on the cost of storing a book digitally.

Dr Lesk explained that the costs in question were for scanning the book versus physically storing the book in a building space in a library.

Professor Katzenelson asked that, regarding code in industry, if there was a requirement that each program had associated with it some pages of text to describe it, which could be searched.

Dr Lesk replied that the requirements were separate from the code, and written independently, if not in parallel to save time. Often the requirements differed from the actual code, causing problems. This could be enforced as a rule but is unpopular with programmers.

Professor Katzenelson suggested that requirements could be searched to find a feature.

Dr Lesk agreed that this would be easy on a text file. He described the problem as the mapping between requirements and code, and outlined proposals to create a very formal mapping between the two, with a line by line association, covering test cases and code modules. He reported that current trials have not been successful.

Professor Randell recalled a management strategy at IBM for hardware design, where new products were required to be made out of a large percentage of existing boards, and secondly that getting a new board into the library was difficult. He emphasised the high quality control of this approach.

Dr Lesk described this practice as being widely used in Japan, where much code reuse took place. Bell Core surveys report this to be cheating, because the different versions are simply for different vendors of the same product. He suggested that often the design of new parts was not necessary because of the existence of parts already, which have previously been unrecorded.

Professor Randell compared this to an old approach of publishing papers, whereby the paper would only be accepted if the author produced the pair of papers that it replaced!

Professor Hall asked if there was a move towards combining precisely fitting parts in text environments, in the same way that reuse was applied in software.

Dr Lesk agreed that the problem of producing complicated hyper-media documents had encouraged such an approach. He also pointed out the problem of quality control during searching, traditional methods implicitly controlled quality by the acceptance for publication. This wasn't the case in Web pages, where quality cannot be evaluated through word searches.

Professor Randell commented that a large number of new services on the Web were aimed at searching and similar facilities.

Dr Lesk agreed, describing the use of selecting movie preferences by the preferences of like minded peers.

Mr Simonyi brought up the subject of his impending talk on Intentional Programming, and outlined the merits of raising the abstraction level of the information. He proposed that by programming using intentions rather than instructions, such word searches would have greater value. He described current programming practice as programming in clichés.

Dr Lesk replied that Bell Core's work was not as ambitious.

Mr Simonyi thought that the work was complimentary, and not similar.

Dr Lesk explained that their work was not at such a high level, and they had not attempted to extract intent. He stated that their main motivation was in condensing code.

Mr Simonyi remarked that in the long term this wasn't important, and that we were merely in a transitory stage, identifying a better way to express the intentions of programmers.

Dr Herbert expressed an interest in the new methods for programming, looking at them from a system management point of view. He suggested that both were involved in finding the information needed to perform such functions more easily. He thought that a major area for the future would be in getting more information about the program into the system, and using it.

Mr Lesk agreed, describing problems with the lack of information available for devices to connect together.

Professor Randell commented that 'plug and play' systems had taken a long time to develop, and drew an analogy with operating systems and hardware, describing the Burroughs operating systems. He outlined the self configuration and device exploration that occurred with the system.

Dr Herbert suggested that the ICL 1900 had similar capabilities, detecting the existence of floating point units.

Dr Lesk said that this approach can't always be used, as algorithms to solve problems may differ, but faster processing chips are beginning to make this unimportant.

Professor Randell asked the final question, bringing up the topic of data mining. He wondered if any work in this area was relevant to the discussion.

Dr Lesk replied that he didn't really know enough about it. He explained that Bell Core's work was very specific. He briefly mentioned the problems of failures of phone networks.

Professor Randell thanked the participants and brought the session to a close.