The EUGRAM, furthermore, has all the advantages of digital storage and accessibility to archiving, sorting and searching mechanisms that are far easier to implement, and require far less bandwidth than do voice messages. The EUGRAM itself can be composed quickly with a text-editor on the user display, where it is readily rehearsed, corrected and re-edited before being transmitted. The same EUGRAM can be fanned out simultaneously to a large number of recipients, or it can be revised and perfected through several versions with similar broadcast, or with selective distribution.

From the receiver's perspective, he has the advantage of a literate spatially oriented medium. In contrast to the time-fluent telephone, radio or TV, he has the option of perusing his mail at his own pace, of interruption, backtracing and cross-checking the text, even of marking it for reexamination and further rumination. He retains mastery of the use of his own time, and can coordinate attention to a coherently chosen set of tasks. He is liberated from the tyranny of synchronizing his own mental processes to those of the external actor. This freedom of course reduces the impact of that actor, just in proportion to the responsible autonomy it returns to the reader.

In framing responses, entire messages or selected extracts together with added comments can be forwarded to others, or returned to the sender -- lending focus to a 'discussion' and providing unambiguous texts for the development of a consensus. EUGRAMs can be filed and retrieved efficiently, or transcribed into hard copy as required. Text editors may be embellished with elaborate formatting aids, spelling correctors, even an online thesaurus to aid in composition. When quantitative calculations are in question, numbers can be mechanically copied directly from program outputs, avoiding pestiferous typographical errors. The same computer is likely to be the user's research tool and give access to shared data-bases: the EUGRAMs can then refer to common files by names that are themselves machinable. The user will also have access to other conveniences, such as desk-calculator-like programs for the checking of figures. He can even track the growing size of a EUGRAM-script (like this one) to be sure it fits into the assigned space. These word-processing capabilities can of course be consummated with hard copy sent through the mails, but with some additional effort, and the degradation of the machinability of the product at the other end.

The paradoxes of instant telephony are most manifest when several parties are involved. In our experience, several weeks prior notice (or other rigid prearrangement) has been needed to schedule teleconferences if four or more people were required simultaneously. EUGRAMs to groups are sent in real time supported by conveniences like group labels. Stored in the receiver's file areas, EUGRAMs are exchanged among an active community like SUMEX-AIM within a few hours, often within minutes. Users also remain in ready communication with each other, via their respective EUGRAM files, even when either or both have travelled away from their customary homes. Lightweight, portable terminals give any user full access to the system from any point which connects to the global telephone and other communications networks. Some facilities offer a fair amount of directory assistance, in locating and identifying the EUGRAM addresses of users; files may also be used to contain blocks of addresses that can be addressed by group names. At SUMEX-AIM, publically accessible bulletin boards are also available for broadcasting information or posting queries, without obtrusion, to a large audience. No doubt, 'junk mail' will become a problem in this medium, as it may in any other. However, the recipient has as powerful a

technology for filtering unwanted messages as the broadcaster has for disseminating them. The struggle is more evenly matched, and there is then less economic incentive for abuses than applies, for example, to the distraction of one's attention by automated telephone sales technology.

Both for the management of the administrative affairs of the system, and for many of the research communications, EUGRAMs have become the preferred method of communication, provided they can be punctuated with occasional formal presentations, and more intimate encounters to help sustain the affiliations of the group. There is still plenty of personal style in the communications, and there is little problem evoking images of the warm bodies at the terminals. This intimacy can and should be supported by encouraging the occasional use of the EUGRAM system for arranging personal rendezvous. The trivial costs of such diversions are more than compensated by the enhanced efficiency of a worker who becomes adept at the use of EUGRAMs as if they were an extension of his own voice or handwriting.

EUGRAMs and Complex Communications [9,10,11]

One of the most controversial questions in social anthropology asks: "Is there a basic difference in modes of thought as between ... 'pre-scientific' and 'science-oriented', 'literate' and 'non-literate' ..." societies [12]. The controversy is complicated by the empirical difficulties of measuring the cognitive styles of individuals independent of their social interactions and of the very media whose effects are in question. The evolutionist would have to interject that a certain neurological development was a precondition for literacy and presumably would have been subject to natural selection at least during the brief interval of human history since the invention of writing. Conversely, the oral tradition made its own demands on other centers in the brain. The only question is whether these cultural patterns have been sufficiently stable and durable to have had a significant effect on the differential evolution of the human brain in different cultures.

Without going so far into the language/thought relationship, we can be categorical about the essentiality of writing for complex cognitive performances. The list -- whether an inventory of baskets or grain, or a city telephone directory -- is an externalization of cognitive activity that invites and sustains public use and scrutiny, and a form that has no effective analogy in the oral tradition. Indeed, it may have been the initial technological breakthrough in record-making preceding other forms of literature. A glance through the pages of this journal is evidence enough of the impossibility of assembling complex scientific arguments without the use of the written record. The manipulation of recorded symbols is a pale shadow of an internal cognitive imagination we hardly understand, but our most intricate intellectual exercises rely heavily on those external marks.

In many instances, it might still be possible to read a journal article over the telephone and garner some degree of comprehension of the argument even without visible records: but consider how often we have to ask simple names to be spelled out and numbers repeated in phone discourse. Imagine then communicating a computer program of more than ten instructions over the telephone! Indeed, it is precisely for the sharing of such program source texts that EUGRAMs have been

most manifestly indispensable for groups like the ARPANET and SUMEX-AIM communities.

These program texts, which may reach hundreds of thousands of instructions are among the most complex records of human logical effort -- and more than any other production, the information is manifestly all in the text. However, they also typify the information content of other scientific efforts like mathematical proofs, structural analysis in chemistry, and other arguments. Some of these also resemble program sources in becoming almost impossible to criticize as written records alone, viz., without exercising them on the computer or in the laboratory. The recent demonstration of the four-color-map theorem comes to mind [13].

One of the facilities offered under SUMEX-AIM is the CONGEN system [14]. This is an aid to the organic chemist, offering him the computer generation of a hypothesis-tree of structures under given constraints. It can also be used as a verifier of claims of new structures, as a proof-checker. As an exercise in advanced organic chemistry, graduate students were assigned the verification of a set of structures recorded in the recent literature. Many of the proofs were found to be incomplete, usually for lack of tacit stipulations that were still plausible in the immediate context. We have no firm statistics, but perhaps one 'proof' in ten contained a substantive fallacy, unnoticed by the author and reviewers, that invited a critical reexamination of the conclusion. This suggests that organic chemical analysis has already become too complex for the existing media, that a significant part of the literature is shaky, and that computer-augmented proof-checking of complex structures should be part of the process of editorial review. The prevalence of statistical fallacies in the biomedical literature, often deeply rooted in careless experimental designs, has provoked much critical comment [15-18]. Certainly, it is responsible for a redoubled waste of resources, in the primary efforts, in faulty policy and practice, and in the further work needed for criticism and rectification.

Probably it is wrong to say that chemistry is so complex; to the contrary this finding is more likely a result of the simplicity and transparency of the logical argument in its proofs, which makes them more amenable to computer emulation. Outside of mathematics, very little scientific reasoning has been subjected to formal analysis and representation. EUGRAM publication now affords the opportunities and incentives to undertake more rigorous formulations both by providing more convenient media for depositing illegible proofs and offering access to symbol-manipulating machines to digest them. Increasingly, hardware engineers will find themselves companions to linguists and philosophers of science [19,20]; they have long since shared profitable joint ventures with formal logicians.

Emergence of the New Literacy [8,21]

The previous discussion declaims how the EUGRAM is a return to literacy, with some new forms and tools. The ease of its alteration saves some kinship of the EUGRAM to the oral tradition, with perhaps less social discipline but more effective tools to ensure the authenticity of the text. In fact, so much 'writing' is produced these days by dictation, with the most meagre and clumsy post-editing, that these tools may help bring the author closer to the well-

tempered text he intends. Most tools are two-edged: the ease of inserting cliches and of conforming to system-defined formats may also hinder creativity. But this is like agonizing whether desk calculators will frustrate arithmetic skills. Some authors will balk at learning to type -- even with all the facility of error correction afforded by every editor program. They can doubtless look forward within the decade to voice entry of rough texts that can speed up initial composition, and still leave scope for detailed editing. The author who does not interface directly with his own words with a text-display and editor is missing a powerful and precise organ of expression, which has no practical parallel in human communication today. Still, we can hardly surpass our inherent skills, though the wider availability of these compositional tools and challenges in education might help reverse the trend to illiteracy suggested by all recent statistics.

Not every communication will or should be reduced to an unerasable EUGRAM. Lovers will not be deterred, even by the black box, no more than they are by the mails; but other intimate communications -- particularly some of the angrier ones-- are better left to media where expletives can be deleted in hindsight. Even in scientific communication, there may be a place for a potential refuge: "I never said that?" in retrospection, namely to encourage some irresponsible imagination. This opportunity may be vitiated by the relentless accuracy of the EUGRAM, supported by new methods of encoding 'signatures'. Illegible handwritten scrawls will no longer offer a refuge of ambiguity. Nevertheless, while inscribed promises have more standing in court, voice-to-voice confrontation is less amenable to evasion at the moment: the journal-editor will telephone a delinquent author when repeated pleas by EUGRAM have been ignored. Conversely, the poetic imagination may be less hindered in a literate medium than in immediate confrontation with other critical voices. Ambiguous phrases can be left in the record, when they would be challenged in the vocal stream. These very assertions are ones that might be difficult to articulate in a lecture: they reveal mostly how little we know of the uses of different media.

Most of these remarks have concerned EUGRAMs between identified persons. The use of EUGRAMs for communication with archives opens up additional opportunities and foreseeable problems. In our experience at SUMEX-AIM, EUGRAPHY has been indispensable for the division of labor in drafting and criticizing complicated research proposals: 20 people may be closely involved in a product of 250 print pages. We have not secured a good system for tracking and interleaving successive versions, reducing a hairy tree of separate modifications to a coherent final form. Most nearly fatal is a cleanup reformatting that frustrates any simple line-by-line text comparison of deviant versions!

Confusions of this kind in communal refinement of encyclopedic texts can perhaps be ameliorated with further software for documentation control. However, they reflect an underlying difference between EUGRAMs, manuscripts, and unit copying on the one hand, and letterpress on the other. Gutenberg's method lodges the major cost of publication in composing a definitive version of a master template. A side effect of the economic advantage is the focus on that version as a node of the intellectual commitment of the author, and of criticism by others. Communal revision over a EUGRAM network is likely to outpace the reaction time of individual critics: Scientist "A" will be entering his critique of Heisenstein's Field Theory version 1764 when this has already been revised under the influence of "B" and superseded long since by version 1769. The same

fluidity of commitment may be self-aggravating if scientists are then unconstrained in what they enter into the archives, believing that their errors are erasable, and that they must compete for priority with less scrupulous colleagues. The blurring of nodes of publication will also greatly complicate the task of assigning due credit for intellectual innovation, although in principle there can be greater technological support (auxiliary files and the like) for documenting the participation of many minds. The advantage of this fluidity is, obviously, a possible mitigation of prejudice and rigidity of beliefs that may otherwise impede intellectual progress.

The cost of nodal entry into letterpress also bolsters the gatekeeping role of editors and reviewers as trustees of the social investment entailed in that form of publication. This has already been eroded by the multiplication of commercial interests in scientific journals who receive an large unacknowledged subsidy a) in the public funding of the underlying research, and b) in the asset of attention of the readership. Both of these have been exploited to the point that existing publication is fragmented to an untolerated degree: namely, in many fields scientists no longer accept the responsibility for awareness of every claim that has reached print, particularly if these have bypassed the recognized, peer-refereed organs of their discipline. Near-zero-cost entry into the archives of a EUGRAM system will aggravate that problem, but has the compensation of an easy technology for selective retrieval. The role of the trustees will be shifted from controlling what enters the archives to that of organized consultation about what is worth perusing. Controversial innovations may be more fairly evaluated if minority approval is enough to permit them to reach the visible record.

The same technology can also be used to broaden the participatory base, and to reduce the grievous time lags and enhance the limited information flow that now characterizes peer review of research proposals used for the allocation of budgetary resources. The pros and cons of a wider base of 'voting' on one's colleagues' efforts can be roughly anticipated: in some sense more equitable distributions on the one hand; on the other, the factionalization of decision-making, political alliances, and the tyranny of the majority even in the most creative of individual activities. These dilemmas face us today; the new technologies will introduce a change of scale not of principle in the social monitoring of private thought. It is not just Big Brother we may need to fear, but the whole brood of our competing siblings.

The enemy may also be within ourselves. Scientists generally are systematically socialized within the norms of their profession; nevertheless they must approach the raging floods of literature with some ambivalence [22]: there might be found the nuggets of insight that may help the investigator take a bold new step. There is also the fear of finding an anticipation that may destroy the novelty, and hence the entire utility, of months, years or decades of sweat and the pride of unique intellectual accomplishment. The designer of information systems can ill afford to overlook Mooers' Law, that a "system will tend not to be used whenever it is more painful and troublesome for a customer not to have information than for him not to have it." [23]. Some writers tend to be egregiously neglectful of citing the roots of their ideas, a self-serving amnesia that also obscures others' access to the overall picture. The neglect also impedes the efficient retrieval of connected knowledge through devices like citation indexing. EUGRAM-based commentaries should facilitate the filling in of

missing references by others, if the author has overlooked them, without making a major issue of the implied criticisms; and the anticipation of such corrections may deter the obliteration of the history of a subject. The cross-referencing and coding capabilities of bibliographic databases should also make it feasible for an author to exercise his historical responsibilities without excessively costly footnotes that may impede other uses of the entered material. In a similar vein, the systematic archiving of informal communications, including notes to oneself, surrounding the genesis of new ideas should facilitate the accurate reconstruction of the history of scientific discoveries -- narratives that today are inevitably clouded with more retrospective myth than documentable substance.

Altogether, we simply need to recognize that the new technology imposes fewer constraints per se, on the social structure of science, and that carefully designed new forms of social discipline will need to be established to meet the indicated functional needs.

The social innovations will doubtless evolve in response to microscopic pressures rather than as part of a system design, and their functionality will probably be tested on a time scale slower than continued technological inputs. Some of the needs and inventions can be foreseen; their main effect may be to facilitate another wave of illiteracy by the recruitment of still more elaborate devices for the human-bit interface. Reading and pecking are slow, and beneath the dignity of some professionals; voice response is even cheaper than the visual EUGRAM, and the technology for voice entry is on the way. Graphics already are an indispensable aid; there is no technological barrier to the integration of multi-modal cable-TV (e.g., animated cartoons) with EUGRAPHY. Programming costs will return the initiative to the centralized broadcaster; hopefully, a few individuals will still insist on their own selection of intellectual fare and many will sustain bilateral conversation. The literate tradition can still be enhanced with improved designs of orthographic display, a wider menu of formats including color, perhaps even new alphabets and languages. Indeed, it is language itself that needs more constructive as well as descriptive investigation: our existing tongues have evolved in response to long outmoded technologies of communication, but we know too little of the underlying neurobiology to be confident how they might be improved. Such studies are also impelled by the prevalence of pathologies of language development that constitute a heavy burden on many children and their schools. A 26-character alphabet certainly bears no relationship to any system that would be systematically designed to enhance the speed and reliability of human communications [24,25].

This discussion has intentionally focussed on the difficulties and side effects that may attend the introduction of challenging new technologies of communication [8,26]. Surely others will emerge as difficult to foresee as the impact of the internal combustion engine on the structure of cities. The problems should not obscure the constructive implications of steps towards the realization of an effective 'world brain', which had already obsessed Leibniz, and which may be the defining attribute of technological culture: the efficient refinement and sharing of human knowledge [27]. We do well to question our moral capability of enjoying the fruits of such cooperation; but this is not to damn ourselves in advance, especially if we acknowledge that anticipating the human problems is a task of equal priority to engineering the hardware.

Acknowledgment

The work at Stanford University, on which this essay is based, has been supported by the Biotechnology Resources Program, Division of Research Resources, National Institutes of Health (RR-00785; RR-00612) and by the Defense Advanced Research Projects Agency (Heuristic Programming Project, directed by Professor E. A. Feigenbaum). Many more people have contributed to the ideas presented here -- by publication, by personal interaction, by telephone, and by EUGRAPHY -- than I have been able to acknowledge in the available time and space.

References

[1] F.F. Clasquin and J.B. Cohen, "Prices of Physics and Chemistry Journals," Science, vol. 197, pp. 432-438, 29 July 1977.

[2] President's Science Advisory Committee, "Science, Government and Information," Washington: U.S.G.P.O., Jan. 10, 1963

[3] E. C. Anderson, J. B. Arnold, C. Emiliani, W. H. Johnston, R. L. Miller, H. E. Suess, and V. L. Telegdi, "Back to the Homeric Tradition." American Scientist, 1977, 43:117-119.

[4] J.R. Cole and S. Cole, "Social Stratification in Science," Chicago: Univ. Chicago Press, 1973.

[5] D. Crane, "Invisible Colleges," Chicago: Univ. Chicago Press, 1972.

[6] W. van den Daele and P. Weingart, "Resistance and Receptivity of Science to External Direction: The Emergence of New Disciplines under the Impact of Science Policy," pp. 247-275 in G. Lemaine et al.,eds., Perspectives on the Emergence of Scientific Disciplines, Chicago: Aldine, 1976.

[7] I. de Sola Pool, Ed., "The Social Impact of the Telephone," Cambridge, Mass.: The MIT Press, 1977.

[8] I. de Sola Pool, F. W. Frey, W. Schramm, N. Maccoby, and E. B. Parker, Eds., "Handbook of Communication," Chicago: Rand McNally, 1973.

[9] J.C.R. Licklider, "Libraries of the Future," Cambridge: M.I.T. Press, 1966.

[10] M. Turoff and S. R. Hiltz. "Meeting Through your Computer: Information Exchange and Engineering Decision-making are Made Easy Through Computer-Assisted Conferencing," IEEE Spectrum, May 1977, 58-64.

[11] G.W. Arnold and S.H. Unger, "A Structured Data Base Computer Conferencing System," National Computer Conference, pp. 461-467, 1977.

[12] R.Horton and R. Finnegan, "Modes of Thought," London: Faber and Faber,1973.

[13] K. Appel and W. Haken, "The Solution of the Four-Color Map Problem", in Scientific American, Oct. 1977.

[14] R.E. Carhart, S.M. Johnson, D.H. Smith, B.G. Buchanan, R.G. Dromey and J. Lederberg, "Networking and a Collaborative Research Community: A Case Study Using the DENDRAL Programs," pp. 192-217 in P. Lykos, ed., Computer Networking and Chemistry, Amer. Chem. Soc. Symposium Series No. 19, 1975.

[15] N.D.W. Lionel and A. Herxheimer, "Assessing Reports of Therapeutic Trials," Brit. Med. J., No. 3, pp. 637-640, 1970.

[16] S. Schor and I. Karten, "Statistical Evaluation of Medical Journal Manuscripts," J. Amer. Med. Assn., vol. 195, pp. 1123-1128, 1966.

[17] T. Dalenius, "Bibliography on Non-Sampling Errors in Surveys," Internl. Statis. Rev., vol. 45, pp. 71-89, 181-197, 303-317, 1977.

[18] I.J. Good, "Statistical Fallacies," in International Encyclopedia Social Sciences, New York: Macmillan, 1968.

[19] C.W. Churchman and B.G. Buchanan, "On the Design of Inductive Systems: Some Philosophical Problems", British Journal for the Philosophy of Science, vol. 20, pp.311-323, (1969).

[20] J. McCarthy, "Epistemological Problems of Artificial Intelligence" in Proceedings of the 5th International Joint Conference on Artificial Intelligence, MIT (Cambridge, Mass.), 1977.

[21] J. M. Nilles and F. R. Carlson, Jr., P. Gray, G. J. Hannenman, "The Telecommunications-Transportation Tradeoff: Options for Tomorrow," New York: John Wiley & Sons, 1976.

[22] R. K. Merton, "Sociological Ambivalence," New York: Free Press, 1976.

[23] C.N. Mooers, "Mooers' Law, or Why Some Retrieval Systems are used and Others are Not," American Documentation, vol 11., p. ii, July 1960.

[24] C. Cherry, "On Human Communication," New York: Wiley, 1957.

[25] W.A. Rosenblith, "Sensory Communication," Cambridge: M.I.T. Press, 1961.

[26] G. Gerbner, L. P. Gross, W. H. Melody, Eds., "Communications Technology and Social Policy," New York: John Wiley & Sons, 1973.

[27] M. Kochen, ed., "Information for Action: From Knowledge to Wisdom", New York: Academic Press, 1975.

Appendix III

COMPARISON OF MAINSAIL AND PASCAL

Clark R. Wilcox
Stanford University

MAINSAIL and PASCAL have been developed for different reasons, and it is this which is responsible for the major distinctions between them. The development of PASCAL was based on two principal aims, as stated by Wirth in the PASCAL USER MANUAL AND REPORT (henceforth referred to as the PASCAL REPORT):

> "The first is to make available a language suitable to teach programming as a systematic discipline based on certain fundamental concepts clearly and naturally reflected by the language. The second is to develop implementations of this language which are both reliable and efficient on presently available computers."

The basic goal of MAINSAIL, on the other hand, is to provide a machine-independent programming system suitable for the development of large, portable programs. PASCAL is a sparse, relatively simple language which is really more of a language kernel than a complete programming system. MAINSAIL is broader in scope, requires more runtime support and hence a more powerful processor, but does more for the programmer.

PASCAL as described in the PASCAL REPORT must be characterized as more of a blueprint for a language than a programming system. There are no compiletime facilities, very little standard runtime support, no standard access to a file system, and no concept of module. This lack of completeness, plus the elegance of design of what IS in PASCAL, is the reason for the proliferation of PASCAL implementations (no two of which are identical).

Of course there is no reason why an extended and portable version of PASCAL could not be created, and there has been some work in this area. But this was not Wirth's original goal, has not occurred in practice, and is not the language with which we are comparing MAINSAIL. Such a portable version would presumably be a more complete language, and hence would be a PASCAL-derivative (of which there are many) rather than PASCAL itself.

Portability

MAINSAIL is designed to support portable programs, and as a consequence the compiler and runtime system are largely written in MAINSAIL. The facilities provided by the portable compiler and runtime system are an inherent part of the language. In this sense MAINSAIL is more of a portable programming system than

simply a programming language which can be implemented on many machines. A single compiler and runtime system which are used at all sites appears to be the only realistic means of obtaining the goal of portability. A language description such as that given in the PASCAL REPORT has never been sufficient, and there is no reason to believe that it ever will be.

Of course the most important consequence of portability is that programs designed with portability considerations in mind can be moved among implementations without alteration. A concerted effort has been made to guarantee the characteristics of a sufficiently rich programming environment that the programmer will seldom if ever feel the need to utilize machine-dependencies other than those which are inherent to the task being programmed.

Modules

Perhaps consistent with PASCAL's conception as a simple language, it has no concept of module, i.e., a program is a single unit which results from a single compilation.

To preserve machine-independence, MAINSAIL contains its own notion of inter-module communication, i.e., there is no reliance on a machine-dependent linkage system. MAINSAIL programs consist of independently compiled modules which may be executed from any address within memory, i.e., the modules are position-independent. Modules play a dual role as the vehicle for conceptual program modularization, and as the unit which is moved in and out of memory during execution to provide a virtual memory facility.

This precludes combining MAINSAIL modules with program fragments written in some other language, as could perhaps be done with some PASCAL implementations. However, MAINSAIL does provide for embedded assembly language code.

MAINSAIL encourages the view of a program as an open-ended collection of modules whose identity need not be known when the program is written. A flexible system of dynamic linkage allows arbitrary files which contain the executable code for a module to be read into memory and accessed from any other module which has declared the proper interface. Modules may be obtained from individual files, or from runtime libraries which are built and accessed via MAINSAIL system modules. Multiple instances of any module may coexist: an instance consists of shared code and a separate copy of data.

The lack of any such facilities in PASCAL must certainly be viewed as a limiting factor in its scope of applicability. Any but the simplest PASCAL implementation for machines with a small address space must deal with this deficiency, for otherwise large programs could not fit into memory.

Data types

PASCAL provides the standard data types boolean, integer, real, char, and pointer (defined in terms of another type). Perhaps PASCAL's most important contribution to programming languages is its simple yet extremely useful concepts of enumerated scalar types, subrange types, and set types. Operators for union,

intersection, set difference, (in)equality, set inclusion and set membership are provided.

MAINSAIL provides the data types boolean, integer, long integer, real, long real, bits, long bits, string, pointer, address and charadr (character address). The latter two are so-called low-level types described in a later section. Bits and long bits provide 16- and 32-bit vectors which may take part in bitwise operations such as masking and shifting. Strings are described in the next section.

The ranges of integers, reals and chars in PASCAL are implementation-dependent. In MAINSAIL, integers are guaranteed the range provided by a 16-bit word, long integers and reals that provided by a 32-bit word, and long reals that provided by a 48-bit word. Thus the programmer knows what can be counted on, regardless of what machine executes the program.

MAINSAIL has none of PASCAL's user-declared type mechanism. The best one could do to "simulate" such types would be to use MAINSAIL's macro facility to get the effect of scalar types, use integers for scalar variables, and use the data type BITS to get the effect of sets. This is less readable than PASCAL's approach, and does not provide compiletime checking.

There are some draw-backs to PASCAL'S notion of types, but as usual these can be remedied by a more complete specification. The maximum size of a set is implementation-dependent. Also, PASCAL does not provide for input or output of scalar types or sets except via conversion to and from integer.

Strings

PASCAL has no string data type. Instead, it provides the type char (character). Packed char arrays provide fixed-length strings. Assignment of one packed char array to another involves copying all the characters. String constarts (sequences of chars enclosed in single quotes) can be assigned to packed char arrays. The programmer must keep track of the length of a string.

MAINSAIL provides a full implementation of variable-length strings. A string variable is implemented as a string descriptor which specifies the current length (number of characters) and the location of the first character. The characters are stored in a memory area called "string space". Whenever string space becomes full, MAINSAIL automatically compacts it by reclaiming characters which are no longer referenced by string descriptors.

In MAINSAIL, strings may be assigned, compared, concatenated, "substringed" and scanned in various manners. Other examples of system procedures for operating on strings are those for obtaining the length, first or last character; removing the first or last character; appending a character onto the front or end of a string; converting a value to its string representation and vice versa; reading and writing strings (as well as individual characters) from and to files; reading values from a string (as if reading from a text file); and writing values to a string (as if writing to a text file).

Arrays

PASCAL arrays are restrictive in two ways: they must have constant bounds, and they are statically allocated (unless a component of a dynamically allocated record). Aside from the obvious drawbacks of constant bounds, this restriction also has the unfortunate effect that all array arguments to a procedure with an array parameter must have the same constant bounds. For example, it is not possible to write a general-purpose sorting procedure which works on arrays of different bounds.

PASCAL has the concept of PACKED arrays, and the related procedures PACK and UNPACK to convert among packed and unpacked arrays. Packed arrays are presumably stored in a more compact form than usual, e.g., they can take advantage of subrange types to utilize the minimum bits per element.

PASCAL supports array assignment as a full copy of one array to another. Array comparison is allowed only for packed char arrays, which is PASCAL's representation for character strings.

MAINSAIL's arrays may have variable bounds, and their allocation and disposal is completely under user control. An array is implemented as a pointer to an array descriptor, which is a record which gives information necessary to access the array. The array storage itself is in a separate "record" which is referenced from the array descriptor. Array parameter passing, assignment, and comparison involve just the pointer to the array descriptor.

MAINSAIL's Init statement initializes an array with constant values. PASCAL provides no means of initializing an array other than assignment statements.

There are two penalties for MAINSAIL's more flexible notion of arrays. First, the array descriptor, which must be allocated along with the array storage, takes up storage. For small arrays (10 to 20 elements), the array descriptor is almost as big as the array itself, and is thus a significant overhead. Second, the extra indirection through the array descriptor commonly costs an extra instruction per element access.

Records

PASCAL has records as declared objects, as well as records allocated during execution and manipulated via a pointer. MAINSAIL has only the latter, in accordance with its design philosophy of no static addresses. Similarly, PASCAL has arrays of records and arrays of pointers, while MAINSAIL has only the latter. PASCAL's records must be explicitly disposed by the programmer, whereas MAINSAIL provides both explicit disposal and automatic "garbage collection".

PASCAL has record assignment which involves copying all the fields of the record, whereas MAINSAIL has a copy procedure for this purpose. In practice it is quite rare to copy a MAINSAIL record since it is usually just the pointers that are manipulated. However, PASCAL's static records require copying since they cannot be manipulated via pointers.

PASCAL's variant records and MAINSAIL's prefix classes serve a similar role in that both deal with record types (in MAINSAIL terminology, classes) which, though not identical, share some common fields. In PASCAL a single record type is declared which contains the common fields followed by a form of case selection to choose among the remaining "variant" fields. Whenever a record is created, a type constant is given which corresponds to the "tag field" (one of the common fields). This value is used to indicate which variant of the record to create.

In MAINSAIL, the common fields make up the fields of a separate class, say c. Separate classes are declared for each of the variant forms. These classes specify class c as a prefix class, which means that they inherent c's fields as their initial fields. Thus they all have the same initial fields, and the compiler is aware of this. Where PASCAL utilizes a single record declaration which incorporates the variants, MAINSAIL utilizes multiple class declarations with a common prefix class.

MAINSAIL pointer declarations can be more specific than PASCAL's with regard to variant records since in MAINSAIL a pointer is declared as referring to a particular class (which corresponds to a PASCAL variant), whereas in PASCAL a pointer can only be declared as referring to the record as a whole rather than a particular variant. The result is that MAINSAIL can catch some errors during compilation which are not detected by PASCAL.

MAINSAIL allows "unclassified" pointers which do not specify the class of records which they will point to. Such a pointer can point to any class, and thus the compiletime checking is not possible. This form is used as an escape for those situations for which classified pointers are too restrictive. Since PASCAL pointer declarations necessarily involve a type name, there is no way to deal with unclassified pointers.

PASCAL's WITH statement allows one or more pointers to be specified as default pointers over the scope of a statement. Record fields within the statement may be specified without being qualified by a pointer variable as long as one of the default pointers could be used with the field. The default pointers may not be modified within the statement. MAINSAIL has omitted such a facility since the statement interpretation becomes context dependent, the use of variable names the same as field names is restricted, and it is difficult for the compiler to enforce the rule that the default pointers cannot be modified.

Expressions

Unlike PASCAL, MAINSAIL has an If expression. MAINSAIL also provides an Assignment expression which allows the assignment operator to be used in expressions. MAINSAIL allows comparison chains such as "a < b < c" as an abbreviation of what must be used in PASCAL: "(a < b) AND (b < c)".

MAINSAIL's "dotted operators" are an extremely handy abbreviation:

a .op b	is an abbreviation for	a _ a op b
.- a	is an abbreviation for	a _ - a

where "op" is a binary operator such as "+".

PASCAL does not define the order of evaluation of the operands of AND and OR, whereas MAINSAIL guarantees that only as many operands are evaluated as are needed to determine the result. This is a great convenience; for example it is common to want to evaluate b only if a is TRUE in "a AND b", i.e., a serves as a "guard" on b. This is particularly useful in cases such as "WHILE p AND p.link DO ..." which is not so simply written in PASCAL.

Statements

MAINSAIL has an Expression statement which is simply a dotted expression as described earlier. Examples are

MAINSAIL	PASCAL
`i .+ 1;`	`i := i + 1;`
`s .& "abc";`	`PASCAL has no strings`
`b .IOR procBit;`	`put an element into a set`
`a[i,j] .* 18;`	`a[i,j] := a[i,j] * 18;`
`.- k;`	`k := - k;`

The Case statement is similar in both languages, except in PASCAL a scalar type can be used for case selection, while in MAINSAIL an integer must be used. MAINSAIL has the additional capabilities of allowing a case selector to specify a range of values, and to specify a default statement to be executed in the event that no case selector is satisfied. In MAINSAIL an error occurs if no selector is satisfied (and there is no catchall case); in PASCAL the result is undefined.

MAINSAIL has twelve forms of repetitive statements, whereas PASCAL has four. MAINSAIL provides a DONE statement to terminate an iteration, and a CONTINUE statement to continue an iteration. Both of these can be applied to any level from within a nested iteration. PASCAL provides no such facilities other than an unstructured Goto statement. PASCAL's lack of an iteration terminator causes either a redundant statement, an awkward use of Boolean variables, or a Goto statement.

MAINSAIL provides an explicit procedure return, whereas PASCAL does not. Instead a PASCAL function has an implicit variable given by the name and type of the function. At the end of execution the value of this variable is returned as the result. The lack of an explicit Return statement doubtless leads to a reliance on the Goto statement to get to the end of a function.

The Done, Continue and Return statements remove the need for a Goto statement in MAINSAIL. PASCAL's Goto statement utilizes numeric labels, a rather odd choice, especially considering that labels must be declared. The PASCAL REPORT does not define the effect of jumping into a structured statement.

Procedures

PASCAL makes a distinction between procedures, which do not return a value, and functions, which do return a value. MAINSAIL minimizes this distinction by

referring to functions as typed procedures, and allowing typed procedures to be used in a statement (the result is simply discarded).

PASCAL has value and VAR (reference) parameters. A VAR parameter refers indirectly to the argument variable, and hence is subject to the well known confusions which can arise with this mechanism.

MAINSAIL has three parameter-passing mechanisms. A USES parameter is passed the value of its argument. A PRODUCES parameter is not initialized by its argument; instead it returns its value to the argument upon return from the procedure. A MODIFIES parameter combines the effect of a USES and a PRODUCES parameter: it is initialized by the argument, and it returns its value to the argument upon return from the procedure.

MAINSAIL provides OPTIONAL parameters whose arguments may be omitted in a procedure call (in which case zero of the proper data type is passed), and REPEATABLE parameters which may be passed multiple arguments (in which case the procedure is called multiple times, each time with the next repeated argument).

MAINSAIL's GENERIC procedures are another compiletime feature which provide a simple yet powerful means of using a generic procedure identifier in a call to represent any one of several different procedures as distinguished by the parameter types.

Procedures may be nested in PASCAL, but not in MAINSAIL. The division of a MAINSAIL program into relatively small modules, each of which contains relatively small procedures, virtually eliminates the need for further nesting of procedures.

Unlike PASCAL, MAINSAIL supports OWN variables, i.e., variables local to a procedure which retain their value over procedure entry and exit. PASCAL has parametric procedures and functions, while MAINSAIL does not.

File System

PASCAL says nothing about a file system. Instead there is a standard input file, a standard output file, local files and external files. The PASCAL REPORT states:

> "Files may be local to a program (or local to a procedure), or they may already exist outside the program. The latter are called external files. External files are passed as parameters in the program heading into the program."

Presumably the external files are set up by some means outside of PASCAL, but no mention is given of this mechanism. It is left as implementation-dependent, but even then there is no way provided for the program to have any control over the association of external files with file variables.

PASCAL has a file data type, but it is rather restrictive. A file is modeled as a sequence of components, all of the same type. The components can be accessed only sequentially.

Text files, i.e., files declared as type "FILE OF CHAR", require some extra mechanism. Since PASCAL does not define what characters terminate a line, special procedures writeln(f) and readln(f) are provided which write end-of-line or read up to an end-of-line, and the boolean procedure eoln(f) is true when the end of the current line has been reached in file f. Read and write are extended to allow reading and writing of integers and reals from text files (instead of just chars, which are the components of text files).

MAINSAIL considers a file to be a collection of data, on some external medium, that is treated as a unit by the file system (which is not a part of MAINSAIL). Files exist independently of the execution of a program, so that a program can create a file and associate it with a name which can later be used by another program to access the file. Thus unlike PASCAL, files can provide continuity from one program execution to another.

MAINSAIL makes a distinction between two file types: text and data. A text file consists of characters, and a data file consists of any mixture of numeric and bits data. MAINSAIL also distinguishes two methods of access to a file: sequential and random.

A file is referenced in a MAINSAIL program via a pointer that is produced by the file opening procedure. The pointer belongs to one of the classes "textFile" or "dataFile" which are predeclared by MAINSAIL. tty is a name for referring to the user's terminal. ttyRead and ttyWrite are system procedures used for explicit communication with tty. tty may also be "opened" and treated just like any text file, so that it can be used anonymously by a program. In PASCAL communication with the terminal is presumably provided via the standard input and output files.

In MAINSAIL, a command file (cmdFile) and logging file (logFile) are utilized for standard input and output. Normally these are associated with tty, but they may be redirected to any text file by the programmer, for example to get the effect of a batch stream.

MAINSAIL uses the predefined (implementation-dependent) string constant eol (end-of-line) as a line terminator, instead of special procedures such as PASCAL's readln and writeln. MAINSAIL also defines the string constant eop (end-of-page) as a page terminator. This use of characters to delimit lines and pages is more flexible than PASCAL's use of special procedures since it allows these indicators to be part of a string, and hence implicitly manipulated as data.

Compiletime Facilities

MAINSAIL has a comprehensive set of compiletime facilities which are invaluable in the construction of large programs. The only related facility in PASCAL is the ability to declare constants, and the closely related concept of scalar type declarations (a scalar type can be viewed as a structured set of constant declarations). PASCAL's lack of compile-time facilities reflects its

conception as a simple one-module language rather than a tool for building large programs. The following is a summary of MAINSAIL's compiletime features.

MAINSAIL provides compiletime evaluation for constant expressions, i.e. expressions involving only constant operands. A full macro facility provides definition of constants and arbitrary text with optional macro parameters. The programmer can interactively define macros during compilation.

The Message directive directs the compiler to print a message during compilation. The Sourcefile directive specifies a file which is to be compiled as if its text appeared in place of the directive. Thus text which is to be used in several modules, such as a "macro library", can be placed in a single file and sourcefiled from all the modules.

Conditional compilation allows the programmer to specify under what conditions indicated parts of the source file are to be compiled or ignored. Scanning directives allow pages in the source file to be skipped, and provide for explicit termination of compilation of the current file as if the end of the file had occurred.

A facility is provided for automatic utilization of compiletime libraries, which are just files which contain procedure bodies which are to be "compiled into" a number of different modules.

Save and restore directives allow the compiler's symbol table to be saved, and then restored during some other compilation. This avoids recompilation of frequently used "header" files such as macro libraries.

Other directives give the programmer control over the amount of code emitted to check error conditions such as array subscripts out of bounds and null pointers used for field access.

Low-level Features

MAINSAIL provides features which allow a low-level access to the host machine. They are for use only by knowledgeable programmers who need access to underlying representations or who need to intersperse some machine-dependent code with MAINSAIL code. These features are extensively used by the MAINSAIL runtime system, and thus are an integral part of the language. Nevertheless, MAINSAIL provides enough facilities that those described here can usually be avoided.

The data type ADDRESS is provided for representing arbitrary memory addresses. To access individual characters, the data type CHARADR (character address) is provided. A number of supporting features are included which allow addresses and charadrs to be used for access to unstructured memory.

The Code statement allows assembly language to be included as a string constant which is simply put into the compiler's output file. The ENCODE directive and CODED procedures provide additional capabilities for assembly language programming. Of course, modules which contain assembly language are machine-dependent.

MAINSAIL's low-level features allow procedures to be written in MAINSAIL which would otherwise have to be coded in assembly language. They allow almost all of MAINSAIL's runtime system to be written in MAINSAIL (even the machine-dependent parts). Even those parts which must be written in assembly language (e.g., monitor calls) may be included in MAINSAIL modules. PASCAL has no such features, so that it cannot express manipulation of arbitrary memory addresses.

Appendix IV

AIM MANAGEMENT COMMITTEE MEMBERSHIP

The following are the membership lists of the various SUMEX-AIM management committees at the present time:

AIM EXECUTIVE COMMITTEE:

LEDERBERG, Joshua, Ph.D. (Chairman)
Department of Genetics, S331 [through 8/78]
Stanford University Medical Center
Stanford, California 94305
(415) 497-5801

The Rockefeller University [after 9/78]
1230 York Avenue
New York, New York 10021
(212) 360-1234

AMAREL, Saul, Ph.D.
Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903
(201) 932-3546

BAKER, William R., Jr., Ph.D. (Executive Secretary)
Biotechnology Resources Program
National Institutes of Health
Building 31, Room 5B43
9000 Rockville Pike
Bethesda, Maryland 20014
(301) 496-5411

COHEN, Stanley N., M.D.
Department of Genetics and
Division of Clinical Pharmacology,
Department of Medicine, S155
Stanford University Medical Center
Stanford, California 94305
(415) 497-5315

FEIGENBAUM, Edward, Ph.D.
Department of Computer Science
Polya Hall, Room 213
Stanford University
Stanford, California 94305
(415) 497-4079

LINDBERG, Donald, M.D. (Adv Grp Member)
605 Lewis Hall
University of Missouri
Columbia, Missouri 65201
(314) 882-6966

MYERS, Jack D., M.D.
School of Medicine
Scaife Hall, 1291
University of Pittsburgh
Pittsburgh, Pennsylvania 15261
(412) 624-2649

AIM ADVISORY GROUP:

LINDBERG, Donald, M.D. (Chairman)
605 Lewis Hall
University of Missouri
Columbia, Missouri 65201
(314) 882-6966

AMAREL, Saul, Ph.D.
Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903
(201) 932-3546

BAKER, William R., Jr., Ph.D. (Executive Secretary)
Biotechnology Resources Program
National Institutes of Health
Building 31, Room 5B43
9000 Rockville Pike
Bethesda, Maryland 20014
(301) 496-5411

COHEN, Stanley N., M.D.
Department of Genetics and
Division of Clinical Pharmacology,
Department of Medicine, S155
Stanford University Medical Center
Stanford, California 94305
(415) 497-5315

FEIGENBAUM, Edward, Ph.D.
Department of Computer Science
Polya Hall, Room 213
Stanford University
Stanford, California 94305
(415) 497-4079

LEDERBERG, Joshua, Ph.D. (Ex-officio)
Principal Investigator - SUMEX
Department of Genetics, S331
Stanford University Medical Center
Stanford, California 94305
(415) 497-5801

MINSKY, Marvin, Ph.D.
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge, Massachusetts 02139

MOHLER, William C., M.D.
Associate Director
Division of Computer Research and Technology
National Institutes of Health
Building 12A, Room 3033
9000 Rockville Pike
Bethesda, Maryland 20014
(301) 496-1168

MYERS, Jack D., M.D.
School of Medicine
Scaife Hall, 1291
University of Pittsburgh
Pittsburgh, Pennsylvania 15261
(412) 624-2649

PAUKER, Stephen G., M.D.
Department of Medicine - Cardiology
Tufts New England Medical Center Hospital
171 Harrison Avenue
Boston, Massachusetts 02111
(617) 956-5910

SIMON, Herbert A., Ph.D.
Department of Psychology
Baker Hall, 339
Carnegie-Mellon University
Schenley Park
Pittsburgh, Pennsylvania 15213
(412) 578-2787 or 578-2000

STANFORD COMMUNITY ADVISORY COMMITTEE:

LEDERBERG, Joshua, Ph.D. (Chairman)
Principal Investigator - SUMEX
Department of Genetics, S331
Stanford University Medical Center
Stanford, California 94305
(415) 497-5801

COHEN, Stanley N., M.D.
Department of Genetics and
Division of Clinical Pharmacology,
Department of Medicine, S155
Stanford University Medical Center
Stanford, California 94305
(415) 497-5315

DJERASSI, Carl, Ph.D.
Department of Chemistry, Stauffer I-106
Stanford University
Stanford, California 94305
(415) 497-2783

FEIGENBAUM, Edward, Ph.D.
Department of Computer Science
Polya Hall, Room 213
Stanford University
Stanford, California 94305
(415) 497-4079

LEVINTHAL, Elliott C., Ph.D.
Department of Genetics, S047
Stanford University Medical Center
Stanford, California 94305
(415) 497-5813