Workshop on the Icon Programming Language

Ralph E. Griswold

Department of Computer Science
The University of Arizona
Tucson, Arizona

IPD61a
March 14, 1996
http://www.cs.arizona.edu/icon/docs/ipd061.html

A Workshop on the Icon Programming Language

July 26-29, 1988
Flagstaff, Arizona

A workshop related to the Icon programming language took place at Northern Arizona University (NAU) in Flagstaff, Arizona on July 26-29, 1988. Steve Wampler, who initially suggested a meeting on Icon, and who organized the workshop, served as host.

Twenty-eight persons were invited to participate, including the Icon Project group at The University of Arizona and others closely associated with Icon. Fifteen of those persons attended. Their names, addresses, and electronic mail paths are attached at the end of this report.

The workshop was held in a conference-room setting in the Engineering Building at NAU. There was an organizational meeting on the evening of July 26, and there were morning and afternoon sessions on the 27th and 28th. A final morning session took place on the 29th.

The Workshop Setting

The organization of the workshop was designed to be flexible. Some topics for discussion were suggested in advance. Others came up as the workshop progressed. Sessions were informal, consisting mostly of open discussion with only a few "presentations".

Wednesday Morning, July 27

Ralph Griswold opened the first technical session with a review of Icon's origins, its history, and its present status. Discussion turned to the kinds of applications for which Icon is used. Rick Fonorow described Icon's use for prototyping at AT&T Information Systems, listing those aspects of Icon that had been particularly helpful:

The null value (as an initializer).

Quick program development cycle.

Tracing (which was carried over to the C program when the prototype was converted to a production program).

Expressive power.

Data structures.

Automatic storage management.

Lack of size limitations.

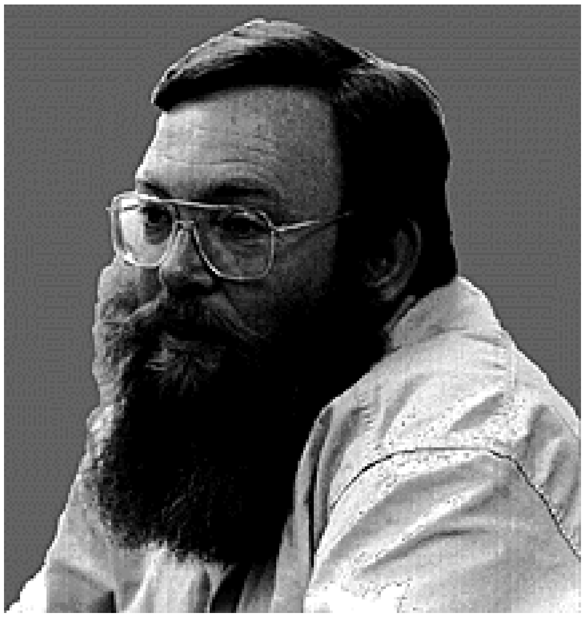
Matters related to corporate reluctance to accept a "non-standard" programming language were mentioned.

Next, Paul Abrahams described his experience using Icon on an MS-DOS system to simulate IBM's generalized mark-up system. He found co-expressions particularly useful and used them extensively, although he noted that all his uses were " one- way " and asked if anyone had a good example of " two-way" use. Steve suggested cellular automata as an application for this. Paul also commented on the usefulness of null records as type markers. He commended Icon for its "superb engineering", including attention to detail. He raised the question of whether it was reasonable to encourage the use of "thousands" of co-expressions in a program. Among the problems he noted were slow running speed (especially the Lattice expandable-regions version), poor error handling in MS-DOS itself, and fundamental problems with the handling of global program structure in Icon. He also noted pattern matching was more difficult in Icon than in SNOBOL4.

Paul Abrahams

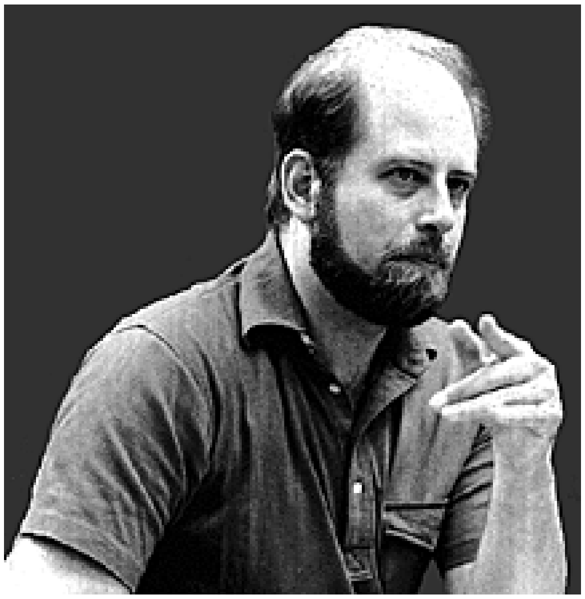
Ralph commented that a lot of work had been done trying to make string scanning more satisfactory, but no really good solution had been found. He suggested that Ken Walker describe earlier work on co-expressions in a subsequent session. Paul concluded his discussion by suggesting three areas of Icon that deserved serious attention: global program structure, pattern matching, and compilation.

Jerry Nowlin next described his experience using Icon at AT&T, primarily in terms of individual tools to facilitate program development. Discussion focused on programmer productivity versus efficient execution, as well as the reaction of programmers who were used to Pascal/C paradigms when presented with Icon. He was especially concerned whether programmers used Icon as a kind of high-level C or whether they used Icon to its full advantage. He also asked for advice on how to advise programmers about ways to write more efficient Icon programs.



Jerry Nowlin

Ralph asked, in this regard, about the organization of the Icon Programming Language book (Prentice-Hall, 1983), which presents conventional topics first and defers generators and string scanning until later. He commented that this matter was topical, since he and Madge Griswold were revising the book and planned to introduce generators and other essential aspects of Icon earlier.



Bob Goldberg

A plea was made for an Icon reference card.

Discussion next turned to the teaching of Icon-why and how. Topics mentioned were:

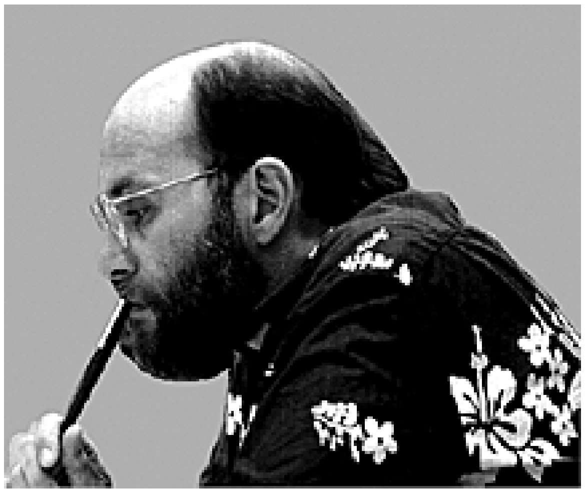
> Rich data structures.
>
> Weak typing (as a contrast to other currently popular languages) .
>
> Easy debugging.
>
> Generators (although it was noted that they required a sophistication that many students lack).

Bob Goldberg commented that the free availability of implementations of Icon was a very important matter in academic environments.

Ralph commented that Icon appeared to be gaining wider acceptance in academic curricula, although Prentice-Hall no longer provided information about adoptions. He also mentioned that finding instructors who are qualified to teach Icon is difficult.

Next followed discussion on how to get Icon more widely known. Bob Alexander suggested that priority be given to revising the overview document to include examples that were more motivating. An issue of BYTE on Icon was also suggested, although BYTE's interest in subjects as "esoteric" as Icon was questioned. Ralph commented that it basically came down to advertising and there was only a limited amount that could be done in Icon's present non-commercial status.

Wednesday Afternoon, July 17



Mark Emmer

The afternoon session began with the topic of possible language enhancements and changes. Paul presented a proposal for a module structure for Icon that would facilitate the handling of global program structure. A module would consist of local identifiers and procedures. Once instantiated, components of a module could be referenced with the module name as a qualifier.

There was considerable discussion of syntactic issues as well as visibility of identifiers in a module. The relationship of modules to object-oriented features was noted, and object orientation was placed on the agenda for further discussion. It was also suggested that a module might provide an entity for parallelism, another topic for discussion. Since many other programming languages have modular features, a study of other languages was suggested.

The next topic for discussion was Bob Alexander's proposal for "iterative conjunction"

```
        all exprl of expr2
```

which would behave similarly to `while-do`, except that *expr2* would not be bounded. The motivation for this control structure would be to provide mutual evaluation of an arbitrary number of expressions (in *expr2*), which presently cannot be formulated in Icon.

Steve noted that this problem comes up in programming *n* -queens. For a specific *n* , Icon generators can be used in a natural way, but when *n* is unknown, entirely different techniques are required.

Janalee O'Bagy noted that this feature interested her, since it filled an existing gap in the matrix of control structures in Icon. That is, there presently is no looping control structure in which both the control expression and the body are unbounded.

Janalee O'Bagy

In attempting to describe iterative conjunction, it became evident that there were conceptual problems with how it really worked. On the other hand, it is easy to implement, so Ralph volunteered to implement it so that it could be used experimentally to see if it really was useful and if there was a simple way of describing its behavior.

Next Ralph mentioned two topics that arose frequently concerning extensions and changes to Icon: sorting and interfacing system routines.

Several possibilities for changes to sorting were discussed, including providing arguments to sort to specify keys and providing a function or procedure to compare values. No proposal seemed to gain general support.

On the matter of interfacing system routines, Bob Goldberg offered Macro SPITBOL's `HOST()` function as an example of how *not* to do it. Again, there was considerable discussion and agreement on the importance of such a feature, especially on computers like the Atari ST, Amiga, and Macintosh, with their enormous repertoires of graphic, sound, and "toolbox" routines. However, no one could suggest a good method for implementing an interface.

Ralph next discussed the issue of preprocessor for Icon, noting that a C-style one written by Kelvin Nilsen

Kelvin Nilsen

The afternoon session closed with Kelvin describing the features of the experimental version of Icon that was the focus of his recently completed doctoral research on high-level programming language facilities for real-time programming. He described streams, which provide a generalization of string scanning, and processes which replace co-expressions as a basis for dealing with concurrency.

Thursday Morning, July 28

The morning session opened with discussion of standardization. Gregg Townsend mentioned reasons why a standard might be useful before there was a further proliferation of implementations. He noted that different implementations of Icon already had different sets of extensions. He also noted a standard's value in precisely describing Icon's semantics for potential implementors. Paul noted that a standard defined a floor, not a ceiling, on a language's features, and there should be no problem with extensions in a particular implementation. Jerry noted the psychological value of a standard in establishing credibility in industrial environments. Ralph suggested additional annotation of the source-code listings as an inexpensive way of getting something useful. He commented that in all such proposals, there was a serious question as to who was going to do it and where the resources would come from.

Gregg Townsend

Discussion next turned to "object-oriented" Icon. Bill Griswold discussed the basic aspects of object orientation: that a value should know what it's all about; that it should have "self awareness" and be capable of being told "do this to yourself". He mentioned various object-oriented paradigms, such as message passing in Smalltalk. He mentioned the relationship of co-expressions in Icon to object-oriented programming. The discussion of object orientation concluded with a discussion of types, subtypes, inheritance, and sharing.

Next, Bill described an experimental object-oriented feature for Icon that he developed using Icon's variant translator mechanism. *Note:* workshop participants were given a handout describing this feature. Bob Goldberg offered to use Bill's experimental features in a class in the fall. Bill noted the similarities between his object-oriented features and Paul's proposed modules.

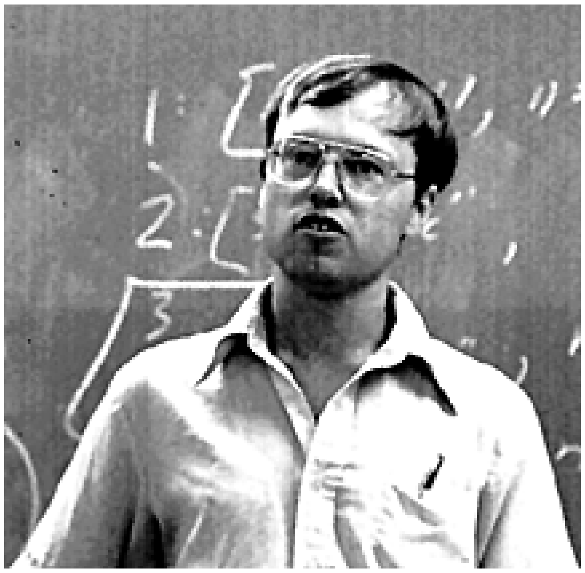
The focus of the workshop then turned to the efficient implementation of Icon.

Janalee described her conceptual models for using recursion in the implementation of expression evaluation. She described how this model could be used to implement a compiler as well as an interpreter. She also described optimizations that could be performed in the context of a compiler:

> Elimination of the costs of bounding expressions in contexts that do not include generators.
>
> The "demotion" of generators in contexts where they cannot be resumed.

She commented that her optimizations would be more significant in combination with knowledge about other aspects of Icon, such as types.

Next, Ken described the type inference system he has developed for Icon. The motivation for this system is that Icon presently must check types constantly during program execution, although most Icon programs exhibit consistent type usage (except for initial null values). His type inference system treats a type as a set of values. He described how he uses abstract interpretation, discarding sequencing information and specific values, to get a conservative but practical assessment of type usage.



Ken Walker

Following this, there was discussion of specific features of Icon, such as the case expression, that might be made more efficient in the interpretive implementation as well as in a compiler. It was suggested that case expressions could be implemented more efficiently where all selectors were constants and unique. Ralph commented that efficiency in Icon programming was often misjudged; that Icon programmers often had misconceptions about how expensive specific features are. He gave storage throughput in a read/write loop as an example. He noted this was relevant to Jerry's earlier question about guidelines for writing efficient Icon programs -- some could be given, but in many cases it is best not to be preoccupied with efficiency during program development.

The morning session on the 28th ended with Ken describing the problem with allocating temporary locations for results during program execution. He pointed out the problem with the present stack model in which results are consumed by operations but, because of generators, those results may be needed in subsequent computations. This leads to copying portions of the stack both inefficient and not really stack-like. Ken then showed the conditions under which temporary values must be kept and he sketched an algorithm he has developed for allocating temporary variables in a register model.

Thursday Afternoon, July 28

The afternoon session began with a discussion of possible hardware related to Icon. Kelvin noted that Icon presented problems on conventional computer architecture because it deals with comparatively large objects.

While language-specific architecture seems to be going out of vogue in favor of faster implementations on conventional RISC architecture, it was noted that storage management was one area that might benefit from hardware. Kelvin mentioned the on-the-fly garbage collection algorithm he had developed for real-time applications of Icon and mentioned recent papers on hardware support for such mechanisms.

Ralph commented that storage management had long been one of the major implementation problems for Icon. He pointed out that Icon's storage management originally had been predicated on expandable regions (`sbrk`), but that facility is increasingly less available. He also mentioned the problem with devising a storage-management system that works adequately over a wide range of computer architectures and memory sizes.

Ralph Griswold

The attention of the workshop next turned to pattern matching, with Ken describing co-expressions, a language feature developed one time in an experimental version of Icon. C-expressions resemble co-expressions (which they replace), but unlike co-expressions, which copy their environment from the procedure in which they are created. A c-expression retains a pointer to the environment for its parent procedure. This allows c-expressions to share variables (at the expense of requiring heap allocation of procedure environments). The "bang" operation also causes a c-expression to generate all its values from the beginning of it result sequence, as opposed to co-expressions, which must be refreshed (copied) to start generation from the beginning. Ken gave an example of the use of c-expressions for writing a recognizer for a context-free language .

Dave Gudeman next proposed an alternative to preprocessing as a way of achieving conditional compilation of Icon programs. His proposal involves a constant declaration and the ability, at compile (translate) time, to detect which arm in an `if-then` expression would be always selected during execution, with both arms already compiled. The question was raised as to how much such compile-time recognition and "constant folding" would be supported. For example, alternation is often used in place of `if-then`. In general, control-flow optimization would be required. In any event, there was support for a constant declaration as part of the Icon language, independent of other considerations.

Dave Gudeman

Bill next described the "extension interpreter" developed as part of a research project on extensibility at the University of Washington. The focus of this work is on the dynamic extension of applications in multiple languages (C and Icon are supported), where the user interface and the (user or library) extensions are factored out from the user application. Dynamic linking was required. In addition, the ability for C to call Icon and vice versa was needed. Although the dynamic linker for C is machine-dependent, the Icon dynamic linker is not. The "CallC" interface is relatively general. The code for it is presently in the central source for Icon, although it has not been tested yet. Other issues are the interface between languages and how arguments of different types are passed and converted.

Discussion next turned to the development of a true compiler for Icon, which is the subject of Ken's doctoral research. Here the "tension between language design and implementation", which Janalee had noted earlier, became evident.

There were comparatively heated discussions on what features might not be supported by a compiler. Steve expressed a desire for identical features in both. Ralph argued that this was impractical and would render a compiler so inefficient as to be useless. The suggestion that string invocation be eliminated in the compiler produced a strong negative reaction from Bill. After much argument, a compromise was drawn in terms of supporting `proc()` but not implicit string invocation. In this scheme, functions would be constants.

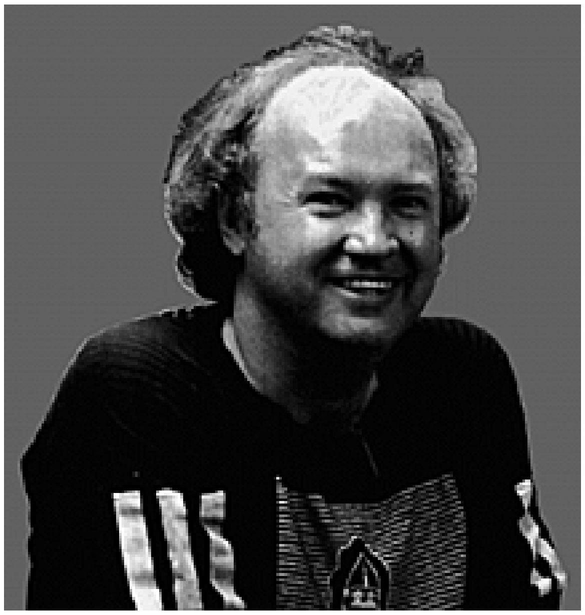
Not much else was decided, except that no one probably would be totally happy with the final results.

Friday Morning, July 29

The final morning was devoted mostly to a discussion of programming environments. Bob Goldberg noted that programming environments are now expected by programmers (as, for example, in the Turbo products).

On the other hand, it is also clear that such environments need to be tailored to the systems on which they run, and a portable design probably is impractical, given the present state of the art in user interfaces.

Bob Alexander commented that he found MPW on the Macintosh a very satisfactory programming environment. Ralph commented that while that may be the case for UNIX-oriented programmers, others expected visual interfaces comparable to those supported by other programming languages.



Bob Alexander

Ralph asked what features one might want in, for example, a debugger. Rick commented he could get along with tracing as it was. It was suggested that tracing of functions, if not operators, be added. Paul suggested the tracing of assignments to variables. The `setexit()` facility of SPITBOL was noted as an effective aid for debugging.

The issue of interpreter/compiler differences was mentioned again. Ralph commented that the same problem existed in compiled languages like C, for which interpreters were subsequently developed. He noted that you can always add interpreter features that are impractical in a compiler and that programmers seem to be satisfied with, for example, using a C interpreter in part of their program-development cycle and then going to a C compiler for production, even though the two have different features. He asked what level of compatibility should be expected between an interpreter and compiler for Icon? Certainly not ucode.

The final topic for the workshop was parallelism. Rick described the interest in Icon for the AT&T 3B4000 and discussed how Icon might be adopted to a multiprocessor system using message passing. Gregg commented that he wasn't sure it was a good idea to add warts to Icon for parallelism. He also commented that generators are inherently sequential and might not fit well into a parallel processing scheme. Bill commented that he had implemented a parallel version of Icon, but that the details probably wouldn't be helpful on another architecture. He also noted that there are very difficult problems, such as garbage collecting in a message-passing environment. Paul suggested parallel co-expressions in which no side effects are guaranteed. Rick again mentioned that the concepts of modules had a very strong relation to parallelism.

Rick Fonorow

The workshop ended with Steve asking how the participants felt about it, whether they wanted to continue on a regular basis, and whether the format was appropriate

Steve Wampler

The participants expressed positive views about the workshop and generally favored continuing it. The possibility of a larger group or a conference with formal papers was mentioned, but most participants felt this would discourage the free exchange of ideas that characterized the current workshop. Tucson was suggested as an alternative and more accessible site.

if it lacked the resources. such decisions would be ineffective. Paul said he thought it was more appropriate for the workshop to discuss issues and provide ideas that would serve as input for the Icon Project rather than render decisions.

The workshop ended on a note of congeniality and with the expectation of future meetings of a similar kind.

Credits

Bill (and friend) did the photography. Ralph scanned enlargements and edited the gray-scale images using Image Studio. He prepared this document using PageMaker and printed it on a Linotronic L-300 image setter. Madge provided editorial assistance and advice.

Participants

Paul Abrahams
214 River Road
Deerfield, MA 01342
Abrahams%Wayne-MTSum.cc.umich.edu

Robert J. Alexander
13856 Bora Bora Way #303-C
Marina del Ray, CA 90292
uunet!dbase!alex

Mark Emmer
Catspaw, Inc
P.0. 1123 Salida, CO 81201
Emmer@arizona.edu
cats!mark@arizona.edu

O. Rick Fonorow
AT&T Bell Labs
1100 E. Warrenville Rd.
IW 2Z-426
Naperville, IL 60566
att!ihlpe!orf

Robert E. Goldberg
4401 W. Estes
Lincolnwood, IL 60646
iitmax edu!goldberg

William G. Griswold
Department of Computer Science
Sieg Hall FR-35
University of Washington
Seattle, WA 98195
wgg@june.cs.washington .edu

Madge T. Griswold
5302 E. 4th Street
Tucson, AZ 85711
madge@arizona.edu

Ralph E. Griswold
Department of Computer Science

Gould-Simpson Building
The University of Arizona
Tucson, AZ 85721
ralph@arizona.edu

David Gudeman
Department of Computer Science
Gould-Simpson Building
The University of Arizona
Tucson, AZ 85721
gudeman@arizona.edu

Kelvin Nilsen
Department of Computer Science
Iowa State University
Ames, IA 50011
kelvin@atanasoff.cs.iastate.edu

Jerry Nowlin
AT&T Bell Labs, IX IC-461
1200 E. Warrenville Rd.
Naperville, IL 60566
att!ihuxy!nowlin

Janalee O'Bagy
Department of Computer Science
Thornton Hall
University of Virginia
Charlottesville, VA 22903
janalee@uvacs.cs.virginia.edu

Gregg M. Townsend
Department of Computer
Gould-Simpson Building
The University of Arizona
Tucson, AZ 85721
gmt@arizona.edu

Kenneth Walker
Department of Computer Science
Gould-Simpson Building
The University of Arizona
Tucson, AZ 85721
kwalker@arizona.edu

Stephen B. Wampler
College of Engineering
Box 15600
Northern Arizona University
Flagstaff, AZ 86011
arizona!naucse!sbw
