Computer Oral History Collection, 1969-1973, 1977

Narrative: Ralph Griswold
Date: May 14, 1972, May 21, 1972, May 28, 1972
Repository: Archives Center, National Museum of American History

This is the beginning of a narrative account of the development of the SNOBOL programming languages by Ralph Griswold. It's my understanding that the material that is being recorded here will be transcribed and returned to me for editing, and that material which is of a sensitive nature will be so marked and will not be released without my permission to the general public. Inevitably, this account is a personal one. The material is narrative, as I have indicated. I have not made any particular attempt to organize the presentation of the material in advance, except to think about the aspects that may be of historical importance and should be preserved. Basically, this material is an attempt to record information, however disorganized, since it is likely to be some time before an organized attempt can be made to record this material. The idea, then, is to make a first attempt to record material that may be forgotten if it is not preserved now.

Any attempt to recapture the history of a project, such as the SNOBOL programming languages, inevitably has many facets. There are, of course, technical ones. There are also important influences in such a project by individuals and by organizations. As I have said, this is my view of the development, and it admittedly carries a strong personal bias. I am attempting to make it as frank and as direct as possible, but it nevertheless would undoubtedly be presented differently by another person from another personal point of view from a different relationship to the project. This first part will consist mainly of a narrative description as I recall it, the development of the SNOBOL languages with asides and anecdotes as I recall them. I'll have some things to say about individuals, but I expect in the later sections to discuss the individuals and their influences separately, essentially as a cross-reference, an illumination of this material. This isn't an autobiography. It shouldn't be. I'm only one of the individuals who have contributed to the development of the SNOBOL languages. But since it is my account, a little background may be helpful in starting out.

I got a B.S. in Physics from Stanford University in 1956. I spent two years in the Navy as a result of NROTC commitment in the Philadelphia Naval Base. I went back to graduate school at Stanford and got my Masters and PhD in electrical engineering. My reason for choosing electrical engineering were primarily my interest in subject matter that could be included in the program in electrical engineering, which at that time was one of the more interdisciplinary and lenient programs. I was not really an electrical engineer; I had no hardware background and don't have any now. I was interested in logic automata theory. My dissertation at Stanford was in switching theory, introducing switching networks. While I had been interested in artificial intelligence topics, brain modeling, logic, and recursive function theory, I had no particular background with computers. I used at one time a 650 that was then at Stanford. I took and simultaneously served as grader for the first programming course at Stanford on the 650 given by Professor Harriet. That so badly turned me off that when I finished my doctoral work at Stanford and interviewed for a job I avoided as much as possible jobs that appeared to be related to computer programming.

At that time, there was a considerable demand for people in computer programming, and a number of people interviewed me with that interest in mind. Again, during interviewing, I was negatively reacting to programming jobs. But I was nominally at least interested in switching theory, although I had become somewhat tired of the subject from doing doctoral work. One of the organizations that I interviewed was Bell Telephone Laboratories and, in particular, by a programming research department (I believe that was the title) headed by C.Y. "Chester" Lee. This was in late 1961 that I was interviewing, around Thanksgiving. How I got connected to Chester Lee's department is somewhat obscure. Other departments which had specific responsibilities for various kinds of engineering development interviewed me, but they for some reason felt that Chester's department might be more attractive to me being a research department. Chester was called in from a closing on his house to interview me and made an impression of some kind on me that eventually caused me to indicate an interest in his area over the others that I interviewed in. He had not talked to me about programming. He talked to me about automata theory; the firing squad problem; switching theory, which was also one of his interests. And as I said, I indicated to the personnel department at Bell Laboratories and other groups that I had interviewed that Chester Lee's was the one that I found most attractive. I think this was somewhat embarrassing to Bell Laboratories, who had probably planned to recruit me into another area, and I didn't understand enough about the structure of Bell Laboratories at that point to appreciate what was going on internally. I do now.

But an offer was forthcoming from Bell Laboratories. I was interested in accepting it, but it wasn't clear to me which area it was in. In fact, I didn't understand the organizational structure of Bell Laboratories to recognize that the offer was, in fact, from a parallel department. And I asked the recruiter if there wasn't a job with Chester Lee. He came back with the answer, "Yes," which I began imagining caused some internal difficulty with Bell Laboratories in the center in which these two departments existed. But in any event, I accepted the offer to come to Chester Lee's department and arrived in February 1962, the first day of work, naturally, being a snowstorm. I went to work in Whippany, commuting from Elizabeth, which is neither here nor there.

I think the first day that I was there I was introduced to Dave Farber, who subsequently played an important part in the development of SNOBOL Languages, who immediately started in his own inimitable fashion instructing me in 7094 Assembly Language BFAB, which I didn't really understand, but nevertheless made a sufficiently memorable impression on me that I had visual recollections of it. It was not at all clear, incidentally, what this department was doing. I was given no specific assignment. I suppose I was supposed to do research. I'm not sure that Chester really cared, whether he thought I was a switching theory type or whether it was just he was simply going to provide an environment in which research could be done. He never provided any kind of noticeable supervision of a technical or personal nature, and I rather suspected his attitude was to higher good people and see what happened.

I did discover, however, that there was work going on in Chester Lee's department on symbolic mathematics, Chester Lee's apparent opinion being that theorem proving could be approached using a computer as a tool, but by way of providing the mathematician a device for symbolic manipulation of mathematical expressions in the context in which he's used to dealing with them, rather than, for example, theorem proving through some heuristic method from a statement of the theorem or through some method of predicate or prepositional calculus. It wasn't at all clear to me what was going on, but Ivan Polansky and Laura Knoll [?], both of whom will be mentioned later, were working on a programming language called SCL, an acronym for Symbolic Communication Language. It was some time before I discovered the history of this programming language, and a lot of it is still not very clear to me. But apparently Chester Lee had designed this language in an attempt to provide a tool whereby the computer could perform manipulations on symbolic mathematical expressions, such as formulas, equations, and had had this implemented. Again, by hearsay, the implementation was done by Dolores Legas [?] in a rather quick, not very sophisticated fashion on the 7094, programmed in BFAB.

Some time earlier, Chester Lee had given Polansky and Knoll the job of using this language, but he had apparently not given them a description of the language. In fact, they complained that they didn't know what all the commands in the language were. And when they found they couldn't do something, they would go back to Chester, and he would say, "Oh, well, there is this command." And then armed with that, they would go back and try again. Over a period of time, they had managed to extract a reasonable understanding of this language, SCL, but they weren't sure that they knew all of it, and there was no documentation, so far as they knew of, that was comprehensive in any way.

At the time I arrived in this group, Polansky and Knoll had been working on SCL VII, I believe was the current edition of it. Possibly, it was SCL V. That's in the literature on Bell Laboratories Memorandum. I believe it was SCL V. They had been using this, discovering how it worked, been attempting to use it for some simple symbolic manipulations. Incidentally, my reason for going into SCL in this depth is it is a precursor of the SNOBOL Languages and was an important part of the context in which SNOBOL was developed. The SCL language deserves, incidentally, a little description. It was implemented on the 7094. It dealt entirely with character strings. There were three data buffers, two of which one could talk about at any particular time by switching focus of attention, I believe. Each of these buffers could accommodate up to 1,024 characters of data as a string. There were pattern-matching statements, the extracting of substrings. There were replacement statements of some kind that permitted reconstituting the strings. Arithmetic was actually performed by constructing a polish prefixed representation of the arithmetic expression. For example, + (3, 5), and then a command was applied to that which evaluated and produced the character string 8, or something like that. It was extremely cumbersome to use, very limited in the amount of data available, and exceptionally slow. Its syntax was rather grotesque, probably bearing strong influences from Chester Lee himself, who as I indicated had an interest in automata theory and switching theory, and is probably closest to being the classical inscrutable Chinese that one could imagine. He could serve as a prototype, in fact, for that. Chester will be discussed in more length later.

But in any event, there was this language and two people trying to use it. It was very limited, but it did provide the ability for symbolic manipulation. It was the only thing available and used by the group at that time. Dave Farber was part of that group also, but not involved in SCL. As a matter of fact, he wouldn't have anything to do with it, as I recall. He felt it was aesthetically disgusting and ridiculous in its concept and implementation. I would comment, going back a little bit, that SCL did have an interesting feature in that recompilation was often invoked during execution to create more programs. In fact, this was necessary to do almost anything and was at least an academically interesting aspect of a language.

I became involved in it rather hesitantly. As I said, I had a negative attitude towards computer programming. I finally managed to get drawn into it. Ivan showed me how to type a job card, which I hadn't done because, I suppose, I was afraid of the computer. I wrote a little SCL program to do heaven knows what. It ran the first time, and it was extremely complicated in its logic since it was recursive in nature. I recall Ivan being amazed at the contorted approach that I had taken, and he found it difficult to understand. I suppose it was partly because it was my first attempt at it and partly because of my background in fields like recursive function theory that caused me without any other knowledge of any other programming language except a little bit of assembly language to approach a programming job in an evolutional way.

Eventually I became more involved in the project and began to work with Ivan at some point on a problem of attempting to factor symbolic polynomials in several variables. That was a rather ludicrous project. It was extremely expensive to run. The runs took so long that they had to be run overnight in the priority scheme of Bell Laboratories. I remember runs that ran two-tenths of an hour that produced half a box of paper because of the recompilation of the program. There was a feature called WOEXP Tag that wrote out the expanded program with tags every time you recompiled. It just produced volumes of paper. These runs were attempting to factor fairly simple polynomials. Now, the SCL language was simply very awkward for doing this kind of thing. Its awkwardness was apparent to us, even though we had no other particular experience. But again, there was nothing else available.

Ivan and I undertook another project, which was the analysis of mark-off [?] chains, which as I recall was an algorithm that we developed that was sort of obviously straightforward, I suppose, for determining the ergodic classes and what not of mark-off chains and for determining the period. This was written in SCL. We got it to work. It printed out results with English language-type text saying the following mark-off chain has the following classes and so forth. Chester Lee was particularly excited at this because of the format of the output. That is, it really said things in English sentences. Well, that of course was trivial when you simply had to put the print statements in the program. But for some reason it excited Chester Lee and made us simultaneously wonder about his own sophistication and what his real goals were.

Somewhere along in this period (I forget where), Dave Farber managed to get COMET installed and working at Bell Laboratories. It was rather a kloogy system. And we, especially I, began to use COMET for some of these problems, and found that it had a lot more capacity than SCL, but still was very awkward and difficult to work with. The point of mentioning this is that COMET is also, in some sense, a precursor of SNOBOL and influenced the development of the first SNOBOL languages. Several people were using COMET. As I recall, Polansky did not keep using COMET or other people from the laboratories, some at Murray Hill, whom I could probably dredge up out of my memory. That's not important. In any event, I used COMET considerably for problems manipulating graphs, some for symbolic mathematics. I also remember doing crystal growth problems in growing, instead of crystals, kidney shaped swimming pools and other things. Certainly, my experience with COMET influenced subsequent development of the SNOBOL language.

At some point in the process, SCL was documented. I believe the documentation was going on when I came to the laboratories. The document that describes it, the Internal Memorandum of Bell Laboratories, was never published externally. As far as I know, nothing was ever said externally about it, except verbally. My name is on that document, although my contributions were primarily editorial. And somewhat typical in the thinking of Chester Lee, the authors are ranked in order of their contribution to the project, not to its documentation. The document, as I recall, is hideously obscure and difficult. It does exist. But at least some of our energies were spent in documenting SCL, and I remember a number of very difficult sessions with Chester Lee who was a difficult person at best trying to get the documentation up to our standards, which were much different than Chester's. Anyway, it was in fact documented at some point. And I have a copy of that memorandum.

Also at some point, it became clear, probably because of my outspoken remarks, that SCL V (or whatever it was) was really inadequate for doing the kinds of things that we wanted to do. It couldn't even do trivial problems efficiently because of the time it took and, in fact, couldn't do them at all because there was not enough space available. It seems strange in a 7094 with some 32,000 octal words of memory that one could not have had more than 3K of decimal of data storage. But that was the way the implementation was. Furthermore, the language was cumbersome. It took a very large number of statements to do anything. The arithmetic, which I described earlier, is by present standards ludicrous, almost unbelievable.

So there became a move to design a better language. At this point, Dave Farber entered into the project with more interest. He's always been a creative, inventive person, and this kind of turned him on. And I remember many sessions with Dave Farber and Ivan Polansky and myself working towards the development of the first SNOBOL language. Certainly, we based many of our languages and ideas on our experience with other languages. Dave Farber had quite a bit of experience in programming; I had very little; Ivan somewhat more, but not a great deal. So we sat down to design a language which would provide a better tool for the kinds of symbolic mathematics and graph theory problems that we were working on.

I think one of the early breakthroughs was in developing the idea of a symbolic name to which could be associated with a string. We now think of that as a variable which has a value which has data type string. But in that period, we did not and made a jump from SCL, which had variables, albeit numbered from 0 to 63 (which should be suggestive), COMET, which had a workspace in which attention was focused in a number of shelves on which things could be stored, and we made the jump—it was not very novel, really, considering what was going on with other languages at the time—to the idea of attaching names to strings. And we call these things string variables. Since that time, we've adopted a more conventional description of them.

The concept of indirect referencing, as I recall, was due to Dave Farber carrying over an assembly language concept but something that made SNOBOL a unique language and still this feature stands as an outstanding characteristic of SNOBOL that's different from other languages. The implication was essentially that every data objective in a language was a string. A string could be used as a variable and have another string as a value, and the indirect referencing operator would go one level and use the value as a variable. Hence, it could have its own value and so forth—a kind of an associative memory.

Now, we developed some ideas. I don't remember all of the things that we went through. The input and output was done in a peculiarly idiosyncratic fashion with special string variables Sys-POT and Sys-PIT [?], which have raised eyebrows elsewhere, which were really taken from, I suppose, the current monitor system standing from Systems Peripheral Output Tape and Systems Peripheral Input Tape. And Sys-PPT for System Peripheral Punch Tape. Those were string variables in SNOBOL that had a special meaning.

Somewhere along the line, we developed the concept of success and failure out of pattern matching, I suppose, in which we took a rather unconventional syntactic structure again with a subject in which attention was focused on a pattern, which would be used to match the value of that subject. In this pattern could be string variables specifying something that would match in arbitrary length string of any characters whatsoever, something would match a fixed-length string of a certain number of characters, construction that would specify matching a literal string and a construction that specified the matching of a string of fixed length. As I recall, my contribution to this was to associate with each of these components a name so that if they matched, the matched value would be assigned to that name as a sort of side effect of pattern matching.

This developed gradually. As I recall, the first real language development and excitement came about in our little group of Griswold, Farber, and Polansky in the fall of '62. We went through many syntactic notations and a number of semantic ideas. As I said, the concept of a string-valued variable, the concept of indirect referencing, the concept of pattern matching, and the by-product of assignment to sub-strings, and consequently the concept of success or failure, because pattern matching might not succeed that embedded in the control mechanism were the essentials of the language.

Simultaneously but secretively, Chester Lee was developing his own proposal for a new SCL language. I don't remember the numbers too well. I think that his proposal was SCL VII, and I think we also thought we were working on SCL VII. He proposed to us one day an extension of his SCL language, in which there were many more fields—tags and flags and… It seemed like dozens of fields that you'd specify complicated things in each line. But it was still similar to SCL in its structure—complicated, intricate, difficult to use, requiring many steps to do anything simple. Our immediate reaction was to turn off very badly at his proposal, and I remember we probably showed it. I have, I believe, pictures taken at the blackboard in which he was demonstrating the structure of this language, which should be preserved at some point. But in any event, we rejected his proposal out of hand. Chester was not a person that you could argue a reason with; particularly it was impossible to criticize him without causing him to react violently or to completely shut you out. And we proposed our language to him. This developed into a personality struggle. More of that later.

We were quite convinced that what we were doing was worthwhile, and we continued. At some point towards the end of 1962 or early in 1963, we were convinced that we had enough of a language that we could use, and we decided to implement it. I'm sure that this was influenced by Dave Farber who was an "old pro" in the field who strongly felt that theoretical work in language design had no value unless the results were reduced to practice. The early implementation, in retrospect, is a great curiosity. Dave was an experienced programmer and had done a fair amount of programming work for the operating system, though I'm not sure how much. Ivan had some assembly language experience. He had coated an octal at the Bureau of Standards—literally octal. I think he knew the 7094 assembly language, although I don't think he'd ever programmed anything on it. He at least understood the nature of it. I had had experience in SCL and COMET, and Ivan in SCL. And we sat down to write an implementation.

Now, the structure of this implementation is very heavily influenced by Dave Farber. He was the only knowledgeable one there. Ivan and I wouldn't know how to have implemented a programming language. Dave had a number of ideas: first that it should be an interpreter—would in fact have to be an interpreter—because it was a very high-level language dealing with difficult concepts; that the source language program would be translated into a table of pointers and flags internally, which would be interpreted; there would be a pattern matching algorithm separately; the dynamic storage allocation would be required because of the creation of strings and their use as variables to the indirect referencing operator. Basically, he came up with a scheme of implementation consisting of a compiler, an interpreter that he offered to write), a pattern matching program that Ivan and I were to undertake, and Laura Knoll was assigned the job of doing storage allocation along designs developed by Dave.

Now, Dave was in contact with Doug McElroy, officially known as Malcolm Douglas McElroy (but he doesn't like Malcolm) at Murray Hill. And at the time, Ivan and I were under the impression that Dave was developing most of the implementation design, which we of course contributed our ideas to but did not have much experience. Later, it became more apparent to me that Dave was in fact going to Doug for suggestions. At this point, I do not know to what extent the original implementation of SNOBOL was designed by Doug McElroy or possibly Bob Morris, who was there at the time, or by Dave Farber. As far as we were concerned, Dave was doing it, but I do know that he got a considerable amount of input from McElroy.

The implementation was done quite quickly. We had something running in about three weeks, which was, I think, rather remarkable considering that I had never written an assembly language before and I didn't know the order structure of the 7090. Ivan had to explain those to me, and Ivan was not then and is not now a good coder. Dave wrote the compiler; he sketched it out one evening and came in with it written on the back of an envelope. You know, I have often referred to this in anecdotes and talks that I have given about the compiler written on the back of the envelope. There was literally an envelope that Dave came in with. It was a legal sized envelope, not a personal sized envelope. And it did have a sketch of code written on the back. That envelope has not been preserved, I'm sure. It has been somewhat over-embellished because it's an amusing anecdote. But it was essentially one morning he came in and said that he had the compiler written and waved an envelope full of pencil scratches in front of us. Dave wrote the compiler and the basic part of the interpreter. Ivan and I wrote the pattern matching mechanism, which was essentially another interpretive device that operated on the compile program. And Laura, under Dave's direction, wrote a storage allocation, which used hashing techniques to put the strings in storage. I'm sure that this was strongly influenced by Morris and McElroy, who had worked in this area earlier, and that a large part of it was, in fact, algorithms taken over from them—algorithms and ideas, conceptions. I don't really recall very much about the implementation of this original version of SNOBOL at this point. If there still isn't a listing in existence, I don't at the present time have it, the present time being May 1972. But it may be more likely there is somewhere a binary deck with a system.

Well, anyway, we got it running, and we got some of the bugs out in late winter/spring of 1963. We rather liked what we had I suppose by comparison with SCL and COMET. It was much faster; it was much easier to write in; statements permitted consistent assignments. Statements and pattern matching and possibly replacement statements, and side effects of assigning values to strings. We wrote programs in this language, which had no particular name I might add at this point, that we could not have managed in SCL. Now, all of this was going on, certainly, it wasn't a secret in any sense, but I don't recall that Chester Lee was very much connected in with it; it was just simply done. He was not the kind of a person that would control such an operation, and therefore effectively he withdrew and we did our thing, Dave Farber being probably the most vocal, obvious leader in the project. Ivan and I contributed to the aesthetics of the language from the user's point of view—Ivan contributing algorithms in some cases and I suppose my main contribution was language features, ideas.

Now, the name for this language is interesting. We have since said that we spent more time trying to find a name for it than we did in implementing the language. And in some sense, that's true, although I don't really recall the actual manpower figures. Although Laura was helping, she was essentially doing specific, assigned program jobs and was not part of the main SNOBOL effort. It was Dave, Ivan, and I. At one point, the language was called SEXI [?], an acronym for String Expression Interpreter, which we thought was very cute. Dave Farber tells the anecdote that when he submitted a job in the computer center, he was in the habit of in the comments field writing the name of the processor that he was using and then his name. He submitted a deck across the counter, which had the comment SEXI Farber. And the girl at the counter said, "That's what you think." Well, we had attached the name SEXI for a while. We recall that we're not really designing a language for general use. We hadn't really thought about it. We were developing a language for our own internal use, although I think Dave Farber saw great possibilities because he was connected to the outside world. Ivan and I were not.

There is an interesting critique of SNOBOL at this point (what was to be SNOBOL) by Chester Lee, which I have a negative copy of. It cannot be Xeroxed. And if reconstructed, in type form as best as possible since the original chemical reproduction process has caused some of the pages to deteriorate. It contains on the front of it a letter by Chester Lee criticizing the language to some extent—we had given it to him for criticism. But the interesting thing to us was that he criticized the name SEXI on the grounds that it showed a lack of confidence in our work. Well, I recall at the time thinking that that was hilarious, and that Chester simply had not managed to be very straightforward in his feeling that such a name was inappropriate. And I think we all agreed that it probably would never be released by the laboratories under that name. But this document exists, and I recall it as being one of the funnier moments in the development of the language. As I said, we'll come back to Chester at some point when I talk about people in more detail.

At one point, the language was called Penelope, for no particular reason, but the fact that we happened to like the name Penelope. There was a very interesting meeting in which we presented our results to Chester in a seminar format in which he said that he was willing to co-author the language with us provided we call it SCL VII (which I guess would have been the next SCL or skipping a number for the big increment), but that if we weren't going to SCL, he didn't want to have anything to do with it. Which is, I suppose, kind of a curious position since he hadn't had anything to do with its development. In fact, he inhibited us whenever he could by at least lack of support, not by forbidding us to work on it. We thought that was just fine with us.

Now, the name SNOBOL resulted after a summer, as I recall, in '63 in which a large amount of time was spent in conference rooms with Dave, Ivan, and me, which again, anecdotally we refer to as throwing paper airplanes and shooting paperclips and rubber bands at each other. But in fact we did a fair amount of math searching for a name for this language that we had developed. And we thought of XL-something-or-other standing for Experimental Language something or other. We went through many possible acronyms working with the basic words that related to what the language was all about—strings, language, symbolic, manipulation, et cetera. We did everything but write a program that took all possible words and all possible orders and generate acronyms. Many proposals came up and were rejected.

I assume the responsibility—somewhat regretfully—that I did in fact come up with the name SNOBOL. That struck Doug Farber immediately as a good name. He was given to being enamored with cuteness. It was taken from the phrase String Oriented Symbolic Language, in which the letters can be found in the right order, although it's not an acronym in the conventional sense and some of the letters are taken out of the middle. In fact, it was designed to poke fun at the process of selecting names for languages—cute names and acronyms. We had no idea at that time (at least I certainly did not) that it would become so popular and that I would have to stand up in front of groups of people who had never heard of the language before and talk about something called SNOBOL, which was sort of obviously cute. Nevertheless, Dave was very excited by the idea. Ivan seemed to like it, also. I made some mild protestations, but we were so badly in need of a name that we accepted it. I don't remember when that name was finally chosen, but presumably some time in middle '63.

By this time, a number of people throughout Bell Laboratories were using it, and Dave had sent off binary decks to a couple of places outside Bell Laboratories, and SNOBOL began to become known and was very favorably accepted by programmers who had symbolic kinds of computation—non-numerical kinds of computation—to perform. This was the beginning, really then, of the public development of SNOBOL, the SNOBOL languages, their attributes being that they dealt with non-numeric data objectives entirely with strings. Again, arithmetic could be performed, but it had to be performed on strings, although not in the prefix format, internally, that Chester had developed. It was a very simple language, very easy to learn, very easy to use.

Some time in the fall of '63, I think Dave Farber gave a talk somewhere and talked about SNOBOL for the first time. A binary deck was sent in to share for distribution. Now, there became a problem of distribution, release outside the Laboratories, of this work. Fortunately for the development of SNOBOL languages, the laboratory was relatively unaware of software and had not developed a policy or attempt to protect this kind of thing. Dave took the position that if you got the release for a talk on something to present it outside of Bell Laboratories—and Bell Laboratories has all kinds of official release procedures, did then… Anyway, Dave took the position that if a talk on a software subject was approved through ordinary release channels by the Laboratories, then that implied the approval of the release of the program about which the talk was made. And it was on this rational that the first version of SNOBOL was released outside the Laboratories.

The documentation for public purposes of SNOBOL first occurred about this time in the fall of '63. We wrote a short paper. As I recall, I wrote the majority of it; Ivan helped some. Dave never did write. As a matter of fact, there is still a memorandum number taken out of Bell Laboratories on implementation of the first version of SNOBOL was never completed, taken out under his name. In any event, we wrote this short paper. Dave, again with his personal contacts, arranged its publication. He went to Dick Hamming, who was then editor of *The Journal of the ACM.* Hamming looked it over, was quoted as saying that this kind of thing needs publication. And he had it published in *The Journal of the ACM* within three or four months without any other reviewing procedure. It appeared in the January *JACM* 1964.

There was no question at this time that the first SNOBOL language was a success in the sense that it was used throughout the laboratories. Users were very happy with it. It attracted a considerable amount of attention. And there were many people who became interested in it. The relatively simple, straightforward description of the language in the *JACM* was also applauded by many people, especially people who had a great deal of background in programming or computer science, because it was very easy to understand. We began to get requests for more information about the language and for copies of the system, which was available in the form of a binary deck on the IBM 7090 series.

The implementation, incidentally, had not improved much in the year since the time that it was started and that it began to be known publicly. The implementation had a number of curious properties that are a byproduct of the way it was done. The compiler would compile absolutely any program into something. This was the compiler written on the back of the envelope, so-called. The only requirement was that the input program had an end card. If it did not, then an error message would be generated when the end of the file is reached. Otherwise, absolutely anything would be compiled into something. This was partly the compiler, but mainly the fact that the way the syntax, the language was designed—any string of characters meant something. Now, meaning something should have some qualification. Any string of characters was a legal SNOBOL IV program.

That is, it had a syntactic meaning and could be parsed, analyzed, and a corresponding, internal table to be interpreted, constructed. However, many operations could not be performed when a program was actually executed, which lead to a diagnostic message at runtime. For example, the SNOBOL compiler would happily accept a COMET program; compile it without comment, provided it had an end card. It would not get very far in executing it because all of the dollar signs, and COMET would produce illegal, indirect references in SNOBOL.

The error diagnostics were provided Dave Farber and had the unfortunate property of being cute. For example, the compiler had an error message, something of the form, “Program too large. Scream for help.” Another one was again something being exceeded, “A whisper for help,” was a terminal comment. Even if that appeared to be amusing the first time you got it as a programmer, it certainly wasn’t after you got it several times. And in that sense, the implementation was somewhat embarrassing, at least to Ivan and me if not to Dave.

There was a classic comment that many people remember from the early days of SNOBOL called “Recompile not enabled.” This was the byproduct of a curious design feature in SNOBOL based most probably on the recompilation feature of SCL in which we associated at least in the language design a label on a statement with the rest of the statement in the sense that the rest of the statement would be the value of that label and when compiled would correspond to the statement. Well, this theoretically meant that if you assigned a value to a string during execution and then branched to the label, which was the same as the name of that string that the string assigned should be considered as program and executed. This feature was not implemented in SNOBOL. It was one of those esoteric things that we thought would be nice. It had the rather unfortunate byproduct of making a program data-dependent in the following sense. Any time you assigned a value to a string, if that string appeared as a label in the program, it changed the program. Had recompile “been enabled” (that probably deserves to be in quotes), then the string would have been interpreted as a statement and compiled.

Recompile was not enabled, so if you subsequently branched to that label, you got an error message saying that the language implementation wasn’t completed effectively. But the more serious problem was that a programmer inadvertently might have used the same name for a variable in his program as for a label and consequently destroy his program. Furthermore, since any nominal string could be used as a variable in a program by use of an indirect reference, and specifically since that indirectly referenced string could be read in from data, it was possible to assign values to variables that could not be determined by examining the program, and hence to inadvertently modify the program itself depending on the nature of the data that the program processed. This so-called recompile feature turned out to be a very bad idea in a sense. It was one of those things you discovered after the language was done. And a number of people have made a similar misjudgment in designing versions of SNOBOL for themselves, thinking that this would be a good idea when, in fact, from a practical situation we knew it was not. The enabling of recompile, as it were, was not implemented until the final SNOBOL IV language was complete.

I don't recall a great deal of the period following the so-called release and public documentation of SNOBOL. I remember that we still continued to work on SNOBOL, and at this point we had pretty much forgotten the original motivation for designing SNOBOL—that is as a tool for doing certain kinds of non-numerical programming. But we were then interested in SNOBOL for itself, I think, and for the first time (at least so far as I was concerned) began to become connected with the professional community outside Bell Laboratories and had the first taste of success, as it were, professionally. I don't recall either what the attitude of our supervision was. Chester was piqued, to say the least, that we had not felt that his ideas for a new language were good, that we had gone off and done our own thing, and that that thing had been successful. As I recall, he pretty much ignored us, although I have sort of vague memories that he may have once or twice come with suggestions for modifications to SNOBOL, more along the lines of the language he proposed. But the working relationship with Chester was never very good. None of us understood him very well, and he probably felt misunderstood in turn.

The thing that was obvious, I think, to all of us except perhaps Dave Farber, who already was knowledgeable in programming and programming languages-- I think it was obvious to all of us that SNOBOL had some serious shortcomings. And in spite of the advantage it offered in simplicity and ease of use and ease of learning, that it lacked some things that would make it much more useful as a language. As people began to use it, that became more apparent and became more apparent to us. I don't know exactly at what point we gave serious consideration to the design of the next language in the series, but I suppose it was a gradual change and movement rather than a specific decision initially that we would be doing this.

The most obvious fault in SNOBOL, as I recall, was the fact that it had no built-in functions. It had only arithmetic. For example, it had no way of testing something as obvious as the length of a string. In order to find out how long a string was, you had to remove characters from it by pattern matching and deletion and count as the characters were removed until eventually no more characters remained and the pattern match failed. Now, that is downright grotesque. It's a very poor semantic match between the basic facilities of the language and the way of doing something quite natural. The length of string was, of course, known, but in the system there was just no way of getting at it directly. Hence, perhaps two orders of magnitude of inefficiency was involved in the determining of the length, to say nothing of the awkwardness of the program, which was probably also an order of magnitude or so more than it need be.

These considerations led us to the design of the next version of the language, which was to eventually become SNOBOL III. Initially, it was to be SNOBOL II, which was SNOBOL with built-in functions for doing things that were needed or natural and some improvements to things such as the way that input/output were performed. The details of the development of SNOBOL III, again, escape me to a large degree. But again, it was the three of us—Dave Farber, Ivan Polansky, and myself—who did both the language design and the implementation.

May 21, 1972

This is the second in a series of segments of narration on the SNOBOL IV language by Ralph Griswold.

As I indicated at the end of the last segment, I have relatively little detailed recollection of the development of SNOBOL III compared to the development of SNOBOL. I always felt that I should have more, and I think it's because I, in my own mind, have a feeling that SNOBOL III was such a considerable advance over SNOBOL that there should have been more to its development, when in fact the implementation of SNOBOL III and its public release followed SNOBOL by perhaps a year and a half, and the advances made and language design were not, per se, major ones. All in all, it went a long way towards making a useful programming language, and SNOBOL III was the first really public general acceptance and knowledge of SNOBOL what SNOBOL III was successful language, still is used to a certain extent today in 1972.

But basically, SNOBOL III differed from SNOBOL by adding a function mechanism that provided the ability for the programmer to do certain kinds of things in a more natural, simple way—you could not do it in a natural, simple way in the original version of SNOBOL. The implementation was done over, but again, a considerable amount was taken from the implementation of SNOBOL. And that, again in itself, was not a major thing.

From a language design point of view, SNOBOL III was essentially the same as SNOBOL. The original idea for the implementation of the SNOBOL III had been to remedy the lack of built-in or primitive functions in SNOBOL by adding a number of functions, which would be part of the SNOBOL II system. That was what we originally aimed for—SNOBOL II—the usual incrementing of the generation number to indicate a new language, a generation in a language of similar background. The functions that were to be added included things like size for getting the size or length of the string, predicates for comparing numerical quantities that returned to the null string and introduced that particular aspect of SNOBOL IV, working into the successor failure mechanism to control program flow, and changing of the names of the I/O associate variables to make them more pneumonic to the user and less pneumonic to systems people.

The implementation of SNOBOL III must have pretty well followed directly on the heels of SNOBOL. I don't really have records on that, and I'm relying pretty much on memory. The storage management was essentially the same, although it was elaborated to include something called hyper garbage collection, which was effectively a relocation and compaction of fragmented storage using a secondary random access facility-- Or, no.

Then it'd be random access secondary storage on the machine. But basically, the language was to be pretty much the same. The only data type was a string. Arithmetic had to be performed on so-called numeral strings (that is, strings which are representations of integers) and so forth.

This time, as I recall, I wrote more of the system. I don't really recollect who did the major part of the writing. Under Dave Farber's guidance, we went more towards a macro-implementation using a string macro packaged by McElroy, adding some of our own, mixing it with 7094 code, the idea being to extend the operation set of the 7094 using macro definitions to provide operations that were needed in the implementation of SNOBOL, which would be fairly understandable with people implementing it. Again, it consisted of a compiler, a translator, which created a table of flags and pointers, a la SNOBOL I, and an interpreter that passed over it. This was my second real effort in writing assembly language code. Of course, I had learned something. I know I was working with Ivan Polansky. Although, again, he never created much code. He was a good person to work with. I remember, I guess, the fact that I wrote the good part of the interpreter because of a compliment from Dave Farber that I found tremendously satisfying. He came over to my house one evening, having seen the first programming of the SNOBOL IV interpreter, and said it was the most beautiful code he had ever seen. Well, that is an exaggeration, of course, but it was a compliment well appreciated and at least reminds me that I wrote a part of the code for the interpreter. The storage manager and routines were again written by Laura White (I guess at that time, before she was married).

Now so far I've commented that it was going to be the SNOBOL II, which consisted of SNOBOL I plus some minor modifications plus a new implementation plus built-in functions. We considered programmer-defined functions at this time and decided how they would be done with the conventions for passing arguments and entry points and returns. We decided again, I'm sure on Dave Farber's initiative, that it should be recursive. But we didn't really have any hope of implementing such a facility. In fact, we implemented SNOBOL without recursive functions first. It was called SNOBOL II. It existed briefly aside from our own internal implementation, which is available to a limited number of people. It existed briefly at Project Mac on CTSS.

The fortunate occurrence that happened that really got us SNOBOL III instead of SNOBOL II was the availability of Lee Varian over the summer of '65, who implemented the recursive function mechanism. Now, Lee Varian is another person that's been important to the SNOBOL project over a period of time, and I'll come back and talk about him later as an individual. But it was his programming ability and his energy and hard work that added the recursive function mechanism to SNOBOL III—that is, to make it SNOBOL III rather than SNOBOL II. SNOBOL III was released internally at Bell Laboratories and then later on outside in the fall of 1964.

There's a period of time that I'm not well able to account for at the moment that could be probably determined by going through notes that do exist (and there is a fair amount of documentation from the development of SNOBOL III). But there is nearly a year between the publication of the internal memorandum on SNOBOL III and the records that indicate its external distribution outside of Bell Laboratories. I suspect (and this should be verified) that that year was spent in changing, improving, and debugging the implementation. During that period, SNOBOL III was available internally within Bell Laboratories in some form or another. I know that a great deal more effort was put into the implementation of SNOBOL III than was put into the implementation of SNOBOL I, and that it was implemented and distributed in at least four forms—for the 7090 under FMS and under IBJOB and for the 7094 under FMS and under IBJOB—the difference there being additional index registers and possibly additional operation codes. I have one existing listing of the SNOBOL III system, which is dated in April 1966, probably the last major assembly of SNOBOL III. So from a period of early 1964 and early 1966, the implementation of SNOBOL III was a major concern.

Other recollections that I have of this period are of a tremendous tussle with Dave Farber over getting him to do a part of the program that he said that he was going to do. Dave was rather notorious for undertaking things that he never completed. And I remember a real set-to [?] with him in an attempt to simply force him to complete it by really putting on the line to him that his assistance was necessary and it would never be done without him. I think that the effort that I put into trying to get him to complete the part of the program he said he would write was less than if I had done it myself. As I said, Dave Farber was notorious for this kind of thing. In spite of the tussles we had, there was never really any acrimonious feelings. And Dave did not, so far as I know, resent the pressure that I put on him to finish it. But I do remember that very well.

The public documentation of SNOBOL III came in the form of an article in *The Bell System Technical Journal*, July-August of '66, which was submitted in March '66. So again, there is this time lag of a year, a year and a half, between the first running version of the language and its public documentation. In fact, to users of the language, it's probably like two years. Internally in Bell Laboratories, of course, there was documentation. The choice of the *BSTJ* was made primarily on the fact that they would publish a long article describing a language almost as a user's manual whereas other journals would not be willing to publish such a long article. And SNOBOL III had gotten to be a fairly elaborate language compared at least to SNOBOL in a fair number of built-in functions and the recursive function facility. Furthermore, its documentation was more extensive—perhaps not as easy to read, but to a growingly sophisticated audience, at least to a large part of that audience, much more satisfactory.

I remained a minority of the audience who much preferred the first description of SNOBOL because of its simplicity and naivety, who resented the technical documentation that is typical of programming areas. This is a controversy that I really take to be associated with the fact that as the use of computers has developed; people have come in from areas outside computer science itself without the technical background or experience and have been rather put down by the technical literature. Of course, a lot of the literature is unnecessarily obscure, but computer science has evolved to the point that most people who use computers are exposed to them in some form or another fairly early in their education, although it still remains a problem, especially in a language like SNOBOL, which tends to be used by people in non-technical areas, such as certain areas in the humanities. Nevertheless, the general acceptance of the computer in our society and its involvement in our education and the way that it permeates our culture has made many people more aware of some of the underlying facts. Consequently, the attitude has changed and still is changing somewhat. I rather suspect that the people now who thought so highly of the original SNOBOL document, some of them still maintain that opinion. It is a nostalgic one at this point and, at the risk of coloring the comment somewhat, I think perhaps a reactionary one.

Following the completion, as it were, of SNOBOL III—that is, the completion of its design, the development of a running implementation which, of course, became something of a maintenance and distribution problem—the technical work that followed was done largely by me and Ivan Polansky in the areas of adding functions to SNOBOL III or manipulating data structures. Now, SNOBOL III had in it the concept of external functions—that is, machine language functions would be written separately and added to the SNOBOL III system when it was loaded. The mechanism for actually adding them was something of a kluge. As I recall, the core was scanned looking for a certain configuration indicating that it did exist, a loaded external function. Nevertheless, it worked very well and provided the possibility for extending the SNOBOL III language. Now, this method of extension was not really very helpful except to people with a fair amount of sophistication and motivation. The main advantage of this facility, in my opinion, was the ability that it offered the people who were working on SNOBOL III to experiment with a language without modifying the implementation of the language itself.

Ivan and I worked on tree functions for SNOBOL III, a set of functions (which, again, as I recall, I wrote most of the code for) Ivan and I designed, which created nodes and then performed operations on them to construct, modify trees. This is documented in the literature. The documentation of it is listed in February of 1965. So this work presumably went on in the fall of '64 immediately after the working implementation of SNOBOL III was available. I think it was probably our major effort during that period of time. And a package of functions was added to essentially augment or extend the SNOBOL III language for manipulating with structures—in this case, trees. This seemed to be very promising conceptually. Certainly, it had important influence on the subsequent development of the language. Dave Farber at this point was doing other things, and oh, I'm sure he was aware of what was going on, but did not contribute actively to the design.

Well, obviously, trees aren't the only structure that one can manipulate. The motivation for them was the fact that here was a language for manipulating non-numeric objects. As

I said, I believe in my earlier description of the origins of SNOBOL that we were working with graphs and other structured objects representing them as strings. Well, representing such things as strings is awkward and limited, has many difficulties that can be discussed. This was an attempt to add to a string manipulation language, essentially list manipulation facilities for a specific case to see how it would work out. Now, of course, SNOBOL III had the function mechanism in it, so sub-jobs could be created by functions and manipulated by functions without any extension of the syntax of the language.

Following that, I became interested in adding other kinds of structures to the language. Now, we recognized that this was experimental, that it was never really intended that these should be incorporated into SNOBOL III as part of the language. But this was a vehicle for language experimentation. I wanted to go on beyond trees and wrote a serious of link list functions for SNOBOL III. Here, I think, that the main contribution was not a conceptual one. It was a practical one. And I had nothing novel in the design of these link list functions. They were, I suppose, influenced by what little I knew of IPL. The work on these link list functions is documented in June of '65. So immediately following the completion of SNOBOL III significant effort was devoted to augmenting string manipulation of SNOBOL III with list manipulation facilities as an experiment. There were a number of other functions added. The literature references Glen Manicker's [?] work on adding some character manipulation functions, a random number generator, and so forth. The work that Lee Varian and I did in adding certain special purpose functions. Again, available as libraries of externals.

Again, my memory fails me or tricks me, and I don't have an accurate recollection of when the idea of SNOBOL IV developed. I know that as SNOBOL III became more popular and as we distributed implementations, that we got many suggestions back for improvements, modifications, changes, additions to the language. I was not convinced this was worth doing. We had a number of ideas, some of them resulting from experimental work on data structures and others from the obvious inadequacies of pattern matching. But I recall personally resisting the development of another version of the language because all of this just seemed like warts to me to be added to SNOBOL III and we didn't have anything really significant to offer.

This was a period in which there was internal turmoil in the Laboratories and computer science areas. The third-generation machines were promised. The Laboratories was in the process of making that ghastly mistake to become involved in Multix [?]. And somewhere along in here (again, we'd have to go to other sources to get the exact date), it became clear that there was a significant revolution in computer facilities coming, probably less significant in retrospect than we thought it was at the time. But nevertheless, it was something was going to have a significant impact.

Now, I think the time at which I became convinced that SNOBOL IV should be done was when I got the idea that patterns should be data objects, not a syntactic notation embedded in a language. That deserves a little explanation. SNOBOL III had specific notation for each kind of pattern structure. It was a syntactic notation written in the pattern field. It permitted matching arbitrary strings, strings bound with respect to parenthesis, strings of a specific length, and any particular string. Furthermore, it permitted the concatenation of any combination of these, one after another. It had no method for describing alternation—that is, to match one pattern or another if that failed. There were many suggestions for augmenting the pattern matching facilities. It was a very obvious area, which people had a number of suggestions very often related to their own particular needs rather than to any general language design. But I resisted this personally because I didn't think that it represented any particular breakthrough, just an elaboration, and I didn't feel comfortable with it in the sense that I didn't really feel it was worth doing.

At some point, because I remember driving in to work one morning down West Front Street, I suddenly got the idea that a pattern could be a data object, that there could be operations on patterns to construct other patterns. This would free the patterns from the syntactic constraint that they had in SNOBOL and SNOBOL III, make it possible for the programmer in fact to build his own patterns and then specify them in the pattern field with a pattern matching statement. Now, this resolved in my mind many problems that had remained obscure. I felt that it was a valuable enough concept to justify the consideration of developing another implementation of the language or another version of the language. This was probably late '65. Well, certainly it was after summer of '65.

At that time, Jim Pogue was, as I recall, our supervisor and had been interested in SNOBOL and wanted to become involved in the SNOBOL project. This idea of a new idea for SNOBOL IV with the assumption that we would be able to develop something by way of list manipulation facilities that provides a part of the motivation for the implementation. A second part of the motivation was the imminent arrival of a new hardware for which there was no implementation of SNOBOL. You can look at this in several ways. Either SNOBOL III was going to be implemented for the new hardware or it wasn't. We obviously wanted SNOBOL and I suppose would have implemented SNOBOL III ourselves on the new hardware if we had done nothing else, but we chose to take the rather imaginary position that there would have to be a version of SNOBOL for the new machine and that just implementing the same thing over again was not the best use of our time and abilities and that we would be willing to implement a version of SNOBOL on the new machines provided that our management would permit us to implement a new version of the language, about which we had some ideas that we thought were very worthwhile. Well, this is more of a state of mind than anything. I will come back and talk about Bell Laboratories in another segment on how it fits into the development of SNOBOL IV. Nevertheless, our position, at least superficially, was accepted by our management to the extent that they in fact cared at all, and we set out to design and implement SNOBOL IV.

By this time, with our experience in implementing SNOBOL and SNOBOL III, working an area programming language development that had been largely neglected by other people and working with groups of users that were fairly far from the mainstream of programming—that is, numerical work or commercial work—we had developed some attitudes about the way in which a programming language should be implemented and designed. SNOBOL and SNOBOL III were designed and implemented by the same people and in both cases by a small group at one time or another.

[Track 2]

…was that one could not really development a good programming language by designing it in advance and then implementing the language according to the specifications of the design, that the only way one could determine whether certain language features were suitable or desirable, properly designed in their details or not was to have a running implementation in which to test new language features. Well, maybe this is more true of a novel language, like the SNOBOL languages, than it is of a numerically oriented language. One could certainly argue that PL1 suffered from the approach to design that was used there. But another point is that language design, in my opinion, can be much more creative if there is a working vehicle in which ideas can be tested with comparative ease, and simply because the testing of the facilities will point out specific defects, deficiencies, or details of design that are unfortunate or inappropriate. That's part of it. The other part of it is that being actually able to use it may stimulate ideas about related language design features that you would never get simply from a design point of view because given the fact that you can run something, even a prototype implementation, there were things that would come up which are a product simply of the fact that you can run something on the computer and hence can process more data that will be suggestive. If this weren't true, the computer would not have been such a useful tool in opening new areas for investigation for problem solving, because very often the computer technique itself suggests something that would not be suggested purely on the basis of pencil or paperwork or cerebral contemplation. Well, that was one very definite goal of the language design implementation project. Stated another way, the design and implementation should be an integrated process in which a prototype system was available to test the language design ideas as they went along.

There were other goals, also. Another one was a flexible system, which could adapt itself to new language facilities. Another was a general philosophical point of view that language facilities should not be rejected simply because they are difficult to implement. There are many, many cases which can be sited in existing languages in which concessions have been made to the implementation that has significantly affected the language itself. Now, this is a state of mind. Part of it, of course, is the question of how really hard it would be to do it or whether in fact anybody knows how to implement a certain facility well or in a practical way. Another question is the question of efficiency. For example, SNOBOL languages handling strings of symbols have very definite problems with efficiencies because the machines are generally ill-suited for these kinds of manipulations. The point of view taken by the SNOBOL IV group (which would be identified as Griswold, Polk, and Polansky at this point with Dave Farber having dropped out because of changing interests and other things that he was doing), the point of view of this group, and I suppose I was the main spokesman of this, was that we were concerned with making programming easier, making the job of the programmer easy rather than making an efficient vehicle. This is purely a philosophy. It has good points and bad points. The de-emphasis of efficiency has obvious by-products, and the ramifications of that are evident. One can still see that as a result of SNOBOL IV language, and part of it was simply to work in an area that we felt had been neglected—that is, providing powerful programming facilities without the constraints of economy.

There was some justification for this in the promises of the third-generation computers. These machines were supposed to be many times faster than second-generation machines and to have much larger memories. And with virtual memory facilities that were then being planned within the context—the Multix, for example—the amount of memory space that was available was essentially, practically unlimited. Well, this merely provided a rational for the point of view that we had, and it has been stated from time to time, and we certainly stated these ourselves. We wanted to see what we could do with a language of this type if considerations of size and space and the implementation were ignored, essentially taking a limiting case of the problem. Well, of course, we didn't in fact ignore size and space entirely, but to a large extent, it influenced our design. We did not reject language features because they might be slow to execute or acquire a large amount of program or memory during execution. We rather took the point of view that there's a worthwhile language feature that would make it easier to program a certain problem to solve a particular problem that this was more valuable. And this could be justified in a research environment where the balance between programming efficiency or between problem solving efficiency and running efficiency is quite different than it is in a commercial environment or in a numerical computation environment where the algorithms are well known and the efficiency and speed of solution is a prime concern. Needless to say, this got us into some difficulty and remains a difficulty, as I have said.

At some point along the line, we were also concerned with machine independence. We had had some experience with SNOBOL III, which had been implemented by us at Bell Laboratories on the 7090/94 and distributed to a number of locations, but implemented in such a way that the implementation itself was of little use to a person who wanted to implement SNOBOL III on another machine. It was assembly language with a smattering of macros, which had the flavor of string operations, logically or effectively extending the repertoire of the 7094 to some string manipulation facility but still design for a particular machine. The algorithms were not apparent from the program itself, and there were a number of implementations of SNOBOL III on other machines. In almost all cases, so far as I know, individuals that implemented it worked from a description of the language, such as the *BSDJ* article to develop their own implementation. These implementations were done, as I indicated, by individuals, not by vendors, corporations. There was little interest or support there, except in one or two cases. Typically an individual in a university or research laboratory undertook to implement SNOBOL III for his machine. The difficulty was lack of support, organization, continuity. Some of the implementations were done to a certain extent, for example, by a graduate student who then left and left behind a system that was incomplete, which was not perhaps fully debugged and with no one who knew enough about it to complete the work.

Aside from the fact that not all these systems ran well, they inevitably had dialects because the individual in implementing SNOBOL III would typically add some facility that he thought was worthwhile, omit something he did not feel was worthwhile, or perhaps implement something in a somewhat different way than we had at Bell Laboratories. For example, pattern matching is a relatively involved process. The algorithm was not precisely stated, and therefore, when individuals implemented the pattern-matching algorithm, they often departed to some extent from the algorithm that we used and hence their language worked slightly differently. We became aware of these problems because Bell Laboratories group was associated with SNOBOL III, and not infrequently, users who were having difficulty with particular implementation would come to Bell Laboratories for help assuming, I suppose, that we were responsible for the system that they were having problems with. We felt this reflected unfairly on us and perhaps were overly sensitive about it. But it did influence the process by which SNOBOL IV was implemented.

We looked towards a possibility of a machine-independent implementation. This idea was, I'm sure, not fully formed or vocalized in the early days. It developed gradually as our concept of machine independence developed. The associated idea of portability—that is, the ease with which a system might be implemented on other machines—came about gradually, and I suppose our concept to that still is developing to some extent. But fairly early, the idea was to extend the concept of a macro-language to designing a language specifically for the implementation of SNOBOL IV. We rather thought of it as an extension of a repertoire for a typical machine—assuming that almost all computers in which we were interested would have things like loads and stores and add and subtract and so forth—to add macros to fill out the repertoire omitting from the language used to implement SNOBOL IV any instructions that were idiosyncratic to a particular computer, such as the convert instructions of the 94. We were also looking forward to a change in hardware.

Now, as I said, probably in late summer of 1965, I became convinced that the implementation of a new language, SNOBOL IV, was worthwhile. There was already an interest in further language design and development on the part of Ivan Polansky and Jim Polk, so I had no difficulty selling this idea. In fact, I had been the one who had held back, as I indicated earlier, because I did not feel we had enough really to justify a new language. The first trial implementation of SNOBOL IV was aimed directly towards the goal that I mentioned earlier—that is, of having a system to test out some of the ideas. We had these new ideas at that time of patterns, which were data objects that could be constructed during program execution and some idea of programmer defined data types. I believe that that idea developed shortly thereafter.

Our first implementation was done in SNOBOL III using the external function mechanism of SNOBOL III to test out these ideas. The actual series of machine-language programs that augmented SNOBOL III was done by Jim Polk, who undertook the responsibility for implementing pattern matching in SNOBOL IV. That was a rather major project, and I forget now how we managed to fake this into the SNOBOL III system so far as syntax was concerned. I believe that we either represented pattern matching as a function or had a pre-processor that mapped the beginning ideas of syntax of SNOBOL IV into such a function to be executed in SNOBOL III with these attached external functions. That effort went on in the fall of '65, and we had not really made a complete commitment to SNOBOL IV but we were definitely working towards it. We were not quite sure how we were going to do this.

There was also the influence of the burgeoning Multix project, which was to provide general-purpose time-sharing facilities throughout Bell Laboratories and Project Mac and elsewhere on a GE645. I myself, and I know Dave Farber, were quite opposed to the Multix project and the way that it was conceived. I remember stating early and restating again and again that it was a mistake for Bell Laboratories under the guise of not having enough computer facilities to risk its production computing for the future on a new an untried system, which was obviously extremely ambitious, that it was preposterous, in fact. The internal politics that contributed to the decision to go to Multix can be documented elsewhere. Suffice it to say that as far as I was concerned, it would be a disaster. I didn't recognize the magnitude of the disaster, but we did not feel that it would be a good idea. Nevertheless, we were working in the environment of a developing Multix system with virtual memories, and it looked as if we would have a GE645 as our computer. And we were certainly influenced by the architecture of that computer and its operating system in our early design.

So the fall of '65 was occupied with experimental implementations in SNOBOL III of the ideas that we had for SNOBOL IV. The "first real commitment and beginning of an implementation" of SNOBOL IV was in February 1966. I think at that time we had made the decision not to attempt to work through the Multix project. Now, SNOBOL IV became part of the official Multix project and was listed in Multix for a period of time, but we never had any contact whatsoever with the Multix management. They never asked us for a report. Presumably, it eventually died without any contact whatsoever. It was just there on paper. And to the best of my knowledge, they could have cared less, and we could have cared less.

We did consider the possibility of working directly as part of the Multix project. That would have required our using a teletype or 1052, I guess, a 1050 into CTSS at Project Mac, presumably writing EPL, the Early PL, which I believe generated output for the 645 or 635 (I don't know which). The EPL compiler, I believe, ran on the 7094 through

CTSS. The output tape was then carried I believe to a 635 that simulated the 645 or something like that, an altogether preposterous method of doing anything worthwhile. Furthermore, it would be very difficult for us to get a significant amount of machine time or priority for access to CTSS, and we decided that we would be much better off rather than working with the Multix experimental software that I described to work with a 7094, which was an existing system in which we had good access, which had debugged software that really worked.

So in February of 1966, we made a commitment to begin implementation of SNOBOL IV for real on the 7094. We recognized that we would have to convert it to some other machine. We thought at that time it was going to be a GE645. It turned out differently. But we felt we'd be better off implementing on a machine which we had access to and then converting it. Now, obviously this influenced again our attitudes about machine independence and portability. We knew we were going to have to convert it, which meant effectively writing it in some way in which we could easily convert it. And as a natural, evolutionary result of our earlier work, that was macro-assembly language in which the macros were to become more important than the underlying assembly language.

For purposes of historical note, I began a diary of the implementation of SNOBOL IV as of that date in February 1966 when we made a commitment, at least to ourselves, to go ahead with the implementation. That was the point when the project really started full steam. We really had a goal and enthusiasm. This diary has continued off and on to the present day. The entries in it are not particularly intelligible. They do pinpoint certain dates at which certain things were done and hence provide some kind of correlation, factual information, about progress. The entries do not, however, contain an adequate amount of material about what our feelings about things were, our thoughts on the development of the project and so forth. They're brief entries made in a great hurry, sometimes scribbled. There are periods in which entries are missing for a considerable amount of time, either because the work was very hectic or because of personal concerns or difficulties. The contents of this diary provide some of the information, but, unfortunately, even to me they are in many cases largely unintelligible. Some chronology here derived from notes in the diaries and other notes will give a feeling for how the SNOBOL IV project functioned.

In the winter of '66, after the February decision, I wrote the compiler for SNOBOL IV using a technique which is at this time not particularly novel, but nonetheless interesting, if for no other reason that it actually worked. The compiler for SNOBOL IV was written in SNOBOL III, not using external functions, but simply SNOBOL III procedures. This was organized in terms of procedures which analyzed the source card, constructed trees (I guess, probably, the trees as external functions were used in its implementation), and then code output from these trees. Now, this is kind of a curious way of writing a compiler. The structure of it is not modern or typical of compilers. The simple reason is that I didn't know any better. Nonetheless, it was done in a fairly small amount of time and then hand transliterated from the SNOBOL III procedures into SNOBOL IV macros, with one procedure corresponding to a stream operation SNOBOL IV being translated into simply a macro with a particular structure. Now, hand translation once the procedures were written was done in a matter of a couple of days or a few hours over a weekend, and as I recall, with only a couple of difficulties, it was gotten running and assembled in macro-assembly language by simply mimicking the structure of a program for a compiler in a high-level language and transliterating it into these macro-operations.

One thing I think deserves mention at some point, and I'll insert it here lest I forget it: the original SNOBOL was implemented very quickly in a very sloppy fashion as an interpreter. I don't think the structure of this was particularly novel. It certainly made no lasting contribution to implementation techniques, and indeed, our concern was with the language design, not the implementation techniques. With SNOBOL III, we attempted a much more professional implementation, both by virtue of its clarity of its internal structures, its error messages, its freedom from bugs, its sophistication, and its ability to handle more complicated structures. It was still an interpreter designed after SNOBOL I. We put a great deal more effort into the quality of the implementation, but again, the language design was the main thing. With SNOBOL IV, although the SNOBOL IV language is vastly different than the first SNOBOL languages and represents a tremendous increase in sophistication and power in the number of original contributions, to the concept of programming language facilities.

We also made a similar incremental jump in implementation techniques, and there was a great deal more effort to the nature of the implementation, and implementation in itself began to become an important part of our own technical interests and motivations. I have already indicated that the factors that affected the approach to the combination implementation language design. Some of these together with the new generation of computers and the proposed operating system Multix certainly made us more preoccupied with the implementation itself. This is a gradual change over a period of time from the implementation first being simply a burden to provide a vehicle for actually implementing our ideas in language design, to the eventual position where the implementation was very often of more interest to us than the language itself. I suppose that that derivation evolution is not surprising, but I'm not sure that it's always been obvious to the people who worked on the project or who used the results of it.

The period starting in '66, in which the official commitment was made to implement SNOBOL IV and in which there is, beginning with that period, a considerable amount of documentation in which there is considerable continuity, represents a period in which there were many new ideas developed. I suppose if I sit here and look at some brief sketches of the chronology, I suspect that we were in some sense influenced by simply the concept of the Multix project in that we were perhaps imitating in a smaller universe the idea of developing a novel system with all of its common organization and structure and play acting at organization. What had before been rather undirected or occasionally somewhat peripheral efforts became a concentrated driving motivation which we were far more conscious of what we were doing. I may perhaps be speaking more for myself than the other people in the project, but it became a real project at this point, with goals, organization, higher motivation, and higher ideals. It became a real thing at this point. I suppose we also recognize that SNOBOL III had really in some sense made a significant contribution to computer science and that we were no longer working simply in private, that our work was in some sense a public thing, and that whatever change success has on such an effort was now becoming evident as the SNOBOL IV effort began.

May 28, 1972

This is the third segment in the series describing the SNOBOL IV language by Ralph Griswold.

There is a lot more to the description of SNOBOL IV development than to the preceding versions of the language. The SNOBOL IV language was a much more ambitious language, both in its language features and its implementation. It was approached, as indicated earlier, in much more professional manner, and the result of that language development has led to a language that has been widely accepted throughout the world and by uses of large-scale scientific machines, and it ranks as one of the five or six major languages of this genre. Therefore, it's not surprising that there is more to history and development, being more recent history, it is better known. Hence, in some sense, a narrative on SNOBOL IV is of less interest than its progenitors. But there are a number of aspects of SNOBOL IV language development that are not public knowledge, and certainly in my own personal view and bias, it's different than that of other individuals that have been connected with the project. Since there is so much of the history, it is more difficult to encompass it in a single narrative. What I will try to do is to describe chronologically the important developments and much as I remember of details of the group that was working on it, and then come back and explore specific aspects, such as, for example, contributions of individuals, the effect of the developing technology and hardware and software parallel to SNOBOL IV, the context within Bell Laboratories in which this work was done, and so forth.

As indicated in the earlier segments, SNOBOL IV history is also better documented. There are still in existence brief notes in the form of a diary, listings of a system from the point in its development where 2400 Mark IV Xerox is available, handwritten draft documentation on certain aspects of the implementation from the very beginning, a large amount of correspondence, a certain amount of memorabilia, and as of this point, a fairly large amount of material that has not been classified and organized to the point that it's particularly useful. Once all this material is organized, this narrative will no doubt have to be modified to reflect more factual knowledge and more accurate chronology.

Basically, the chronology is as follows. Following the period in late 1965 in which the concept of the SNOBOL IV language began to develop and in which it became clear that a SNOBOL IV language might very well be a reasonable goal to undertake, there was this trial implementation in SNOBOL III in which a number of features were explored and expanded, and which led to more confidence that such language features were in fact useful, then in February of 1966, the actual beginning of a SNOBOL IV implementation project. This implementation was begun on an IBM 70, 7094, with the full understanding that that machine was leaving, and that it would have to be converted to another machine. It was not done on Multix for reasons indicated earlier. Certainly, the experience with independent limitations of SNOBOL III and the dialects and incompatibilities and maintenance problems that went along with that, together with the fact that we were fully aware that we undertaking implementation of SNOBOL IV on a machine which was not the eventual project machine for the implementation. This certainly led us into a consideration of a machine's independence and affordability. I wouldn't say that any of us had a great of practical knowledge or we didn't have a great deal of theoretical knowledge; rather, and probably, didn't have our ideas very well formulated. I simply don't recall. We did have some practical experience and the work on machine independence and affordability grew up to sort of a gradual thing, paralleling the SNOBOL IV implementation both, and design of the language.

In March 1966, the compiler functions written in SNOBOL III were interfaced with a system. Shortly thereafter, the interpretative structure of SNOBOL IV was worked out. Ivan Polansky and I went through several iterations on that. The interpretive structure of SNOBOL III was not very well organized. It was not clean, neat. It could not, for example, accommodate the negation operator, which reversed the polarity of the failure success mode in a rule. I remember Ivan and I working out in some pseudo macro language, the first attempts at the interpreter. One of the problems we faced was the problem of name and value. Just again, the first tentative ideas we had about whether or not the object was a name or a value depended on the context in which it was created. We tried first an interpretive system in which objects that were simply values such as literals were referenced by a pseudo name, but then this turned out to be a problem of maintaining two stacks or double entries, and then not knowing, in fact, whether an assignment could be made to an object, not knowing whether it was really a name or only a pseudo name, finally leading to the underlying idea that to exist in the present system, where procedures signal on return whether they are returning a value or a name or failure. This led us to a Polish prefix formulation where the program that would have to be interpreted in which the program was represented by name and values were returned by procedures, such as, for example, arithmetic procedures, or in fact a literal procedure.

One of the problems that came up early was the different between strings and patterns, patterns being data objects, some sense that was not very well thought-out at that time. Strings were, however, simpler. A string, of course, could be used as a pattern and this led to the problem of, well, if you can't make two strings, you get a string, obviously that would be necessary in the rest of the language. If you can [catenate?] a pattern with another pattern, however, you get something different, something more complicated, and if you can [catenate?] a pattern with a string, then there's going to be, you also you get a pattern. First thought was to have two kinds of concatenation operators, one for patterns and one for strings. This seemed to be an unnecessary burden on the programmer, but it was considered briefly. Eventually, it was the band and them in favor of making concatenation of a polymorphous operator. That is an operator that did different things, depending on the data types of its operands, although that rather concise description of the situation was not, I'm sure, evident to us at the time we were developing the language.

In April of 1966, part of the interpreter was working. There was no allocator as yet, and the so-called flexes were considered for patterns, flexes being folded strings. I remember going through at least one design consideration in which a pattern would be expanded into all of its components like a tree. In the case of parenthesized components, this amounted to distributing operators to effectively "multiply out". The patterns, however, led to enormously large, unnecessarily large data objects, when we thought in terms of flexes, or folded trees. In March of 1966, the idea for defined data types was formulated fairly close to what it is in the final version of the language, no doubt suggested partly by the concept of the pseudo-variable in TL-I and the earlier work done in extending the SNOBOL III languages to add string and list data types. Also in March of 1966, defined functions were added to the language, being a minor modification of the SNOBOL III defined function in which it was a default entry point in which the local is a part of the function prototype. Along in this period, the interpreter and the pattern-matching scanner being written by Jim Pogue were merged, and I note that this time the compiler was converted to macro-fab. The earlier comment about the compiler function in March of 1966, referred, I guess to a SNOBOL III implementation.

In June of 1966, the concepts of an inner and an outer system were introduced. These concepts have since been abandoned in the description of the implementation of the project, but they were used throughout most of the implementation and still are referred to by other implementers as SNOBOL IV. For some reason, these concepts seem to have had an intuitive appeal. The inner system was the machine-dependent part of the system sub-routines and the outer system was the machine-independent part, the macro language. Physically this constituted two assemblies, one in BFAP for the 7094 for the inner system, strictly in the BFAP language with macros only as convenient, and the outer system being written strictly in macro language where the macro definitions are written in the BFAP but in which at least conceptually it was machine-independent. We were, at that time, thinking in machine-independent terms, machine-independent data objects, and so forth.

At this time also, the error routines were being written, and the allocator was being designed. The allocator in itself has an interesting history. Bob Yates came on the project between March and April of 1966, after having worked on his own List 2 system for a long period of times. Bob Yates is one of those strong-minded people who will be described elsewhere, individually, in a description of himself as a person. He represented some difficulty to us in the project because his concept of the way the system ought to be designed was somewhat different from mine. And he was far more concerned about efficiency of object code routines than I was, and less concerned about the machine independence. Nevertheless, he did design the allocator, which was based on a List 2 allocator that we had designed in conjunction with someone whose name I don't recall at the moment, which was basically a simple allocations scheme, implementing a pointer, and when the area of storage was exhausted the unused items were compressed out and storage was compacted back to leave a free area, and pointers were relocated accordingly. Most of the ideas that are in the present allocations storage regeneration scheme came to the project through Bob Yates. The original scheme was based on the prospect of having a virtual memory available, which was for all practical purposes, as large as one liked, although there were certainly going to be real constraints when the system was implemented.

The allocator design went through several phases, some of which is documented by draft documentation by Bob Yates. The natural variables, as they are now called, decided to keep the so-called print name—that is, the characters for the natural variable—with the storage that the variable required for its value and so forth, to avoid unnecessary paging and a virtual memory, that being different than SNOBOL III. At one point, I almost thought there would be a number of different storage areas, one for each data type so the data type of an object could be determined by the area that it was in. This eventually evolved into a representation of a source language data type in which there was a code, and the descriptor pointing to the object that identified it as data type. The first allocator was written to work from one storage area into another, being strongly influenced by the concept of an essentially infinite virtual memory. That is, there would be a storage area in which allocation was done. When that area was full, or when there was not enough space to allocate more data, a second area would be obtained, presumably by a call on the operating system, and the storage regeneration program would move and relocate from the old region into the new region and then reestablish a base player on the new region and discard the old region. This was actually implemented on the 7094, so that there, it was necessary to set aside to a fixed region. Then the actual allocator was written purely, or at least claimed to be entirely one pass, doing all of the collection relocation and adjustment of pointers at the same time into the new area. There was a deck of the system in the files of Bell Labs, marked, "Do not destroy, ever." I have no idea where that is now. It may, in fact, have been destroyed because at this point, it would be easier to reconstruct the algorithm than it would be to figure it out from the program. But we recognized at some point that we did not have virtual memory, and we couldn't afford to continue to have half of free storage tied up, just waiting for a storage regeneration and Bob Yates wrote another regeneration algorithm, which took several passes, but did it within the same region.

Now getting back to the chronology, primitive functions were added to the SNOBOL IV system in June and July. There was enough of the system to do timing tests in August of 1966. Again, this is the 7094 implementation. The allocator scanner, the interpreter worked at this point, and output routines had been coded. Between August and September, defined data-type mechanism was coded, garbage collection of the allocated data area was being coded, and arrays were being added. In September of 1966, the idea of a deferred pattern definition was introduced, which gave for the first time in the SNOBOL IV language, the same capability that existed in the SNOBOL III language, where so-called back referencing in which a match for a component in a pattern could be used later on in the pattern to look for an instance of the same thing.

In October of 1966, or perhaps late September, we made one of the more significant decisions in departures of the SNOBOL IV project. At that time, Princeton University had installed an IBM 360. I believe it was a Model 40. It had, I think, two 2311 disk drives and several tapes, and with the active encouragement of Lee Varian at Princeton, who had worked on the SNOBOL III project, and who had rewritten to the 360 some of the basic routines, such as pending specifiers and comparing strings and so forth, we undertook to start a 360 implementation of SNOBOL IV. Now at this point, the 7090 implementation was working at least to the point that we could do timing tests and experiment with the various features. Pretty much the outer system was written entirely in macro language, which we hoped or thought to be machine-independent, and we undertook at this point to convert to the 360. Now, I recall that our original intention had been, or expectation, rather, had been to eventually go the GE 645, under Multix. By October of 1966, the ESS project had already broken with the Multix philosophy when they committed themselves to a 360. There are other sources of documentation and the chronology of that. We became more and more certain that Home Bell, where we were, would not go to a 360, although I don't recall when the 645 order was actually cancelled. But here was a new machine, one of the third generation machines. It was made available to us free at Princeton though primarily the good offices of Lee Varian, and we undertook to start the 360 implementation. We knew nothing about the 360, and when we first looked at the assembly language, we hardly believed it. The basing was among the more unbelievable things. We were incredulous, in fact. Princeton knew relatively little about OS at that point. It was in a very early version. Everything was run online at PCP in single partition mode.

There are many anecdotes connected with the implementation on the 360, which probably tell more about the third generation than they do about the development of SNOBOL IV. Princeton was some 35 miles distant; for one thing, which meant it was a substantial trip and a commitment of time to go there. We made trips back and forth through that fall. First, I recall, Al Jones was working with Lee Varian, and they took our macro source and provided vacuous definitions, I believe, something like a load address or something, just to see how it would assemble, and in the first place discovered that it took two hours to assemble in the Model 40. They had put in the definitions call to other macros so the level of the definition could be determined. At that point, the way we had written the system, and the number of the macros, their definitions contained calls to other macros so there were a nest of macro calls. The amount of time it took to assemble macros, the supposed F-level assembly we were working on was fantastic.

There were a number of other things wrong with the system. At one point, we got into a loop in macro expansion because of a missing end M [?] or something on a macro card, and the operator permitted the assembly to generate six tapes full of intermediate output from the assembler, therefore cutting it. There were a number of other amusing, or at the time not so amusing, circumstances. It was run entirely online, and the punch unit that was then being distributed by IBM was rather notoriously unreliable, and in the event of a punch check, the assembler did not form error recovery. That was an unrecoverable error. Running online, SNOBOL IV would assemble for an extended period of time, in the order of an hour or so, before it began to list the output an punch the object data. The punching and listing were alternate line by line, as was generated, and we had a number of experiences in which we got a punch check part way through and consequently lost the entire result of the previous hours or so of work because there was no recovery and no more listing was provided. One particular problem was large areas of data that we zeroed, and the binary, or a hexadecimal zero on the 360 produces a fair number of punches on a punched card, and it soon became obvious that we would not be able to punch out these modules with large areas of zeros because the punch would simply not survive. Amusingly, no one knew how to direct the punched output object modules in the assemblers in anything but the punch. I recall Lee Varian and the assistant people spending an hour or so, trying to figure out how to do it, and decided that it couldn't be done. They eventually discovered how to get rid of the object module altogether, but that was the level of understanding with the 360 software at that point. There were a number of times in which, of course, the machine was down by the time we got there, or it went down when we were there. And there were various accidents that happened in the building. Someone was welding above a mainframe and dropped hot metal on top of it, or which a load of cement was dropped on top of the mainframe as they were working construction to the building over it. 360 at that point, incidentally, was at the Forestall campus in the linear accelerator building, which is probably neither here nor there.

The conversion, however, was quite successful, in spite of the fact that none of us who were working on the conversion at that point, who were myself, Ivan Polansky, and Jim Pogue. None of us had seen the 360 assembly language before. We hadn't seen the software and we had all these problems. We managed to convert the system in a matter of three weeks or a month to the point that it was running after a fashion on the 360. This involved rewriting inner system sub-routines that were machine-dependent and rewriting the macro definitions. We did borrow some code from Lee Varian and had the help also of Al Jones, who wrote the IL interfaces. We did get it working and the conversion went quite rapidly. It took longer to actually get it functioning and get some of those bugs out of that. This gave us our first success of the goals of machine independence and affordability, which had developed with the project.

In that fall, in November and December, the idea for an integer data type was introduced. Now this is kind of curious, I know, in the language, not to have integers, but both SNOBOL and SNOBOL III did integer arithmetic on strings of characters. SNOBOL III, specifically, to do arithmetic, would take a string of characters, invert it from BCD to integer, like be and whatever else had to be operated on, and convert it, perform the arithmetic operation, and then convert back to BCD and create a corresponding string. SNOBOL IV had integers internally distinct in its implementation, to avoid both the time involved in performing arithmetic operations, and the unnecessary data storage of a result of a string, unless that was actually needed. But it wasn't until late fall of 1966 that the integer data type was introduced in the language so that these two objects, numeral strings and integers, could be distinguished by the programmer externally. This gave us the generalized concept of a data type, which the programmer could talk about, introduce the data type function and really firmly committed us to a multitude of data types which were discernible, differentiable, at the source language level.

[Track 3]

It also introduced the concept of data type conversio n explicitly. Near the end of 1966, early in 1967, there were a number of language features added. I think at this time we were working simultaneously on the 360 and the 7094, updating the macro language. We were moving the machine dependencies that were discovered in the process of conversion, and there were several, because we had not really taken the addressing differences into account adequately, the 7094 being a word-address machine, and the 360 being a byte-address or character-address machine. We had some machine dependencies in our program that we were not aware of. We had to do a lot of rearrangement because of the basing problems on the 360 machine, ill-suited for the kind of system we had developed, or conversely, our system was ill-suited for it; it depends on your state of mind. But we were working on both of these, and I generated, at that point, the program which generates the syntax table descriptions, convert tables for the 7094 translate and test tables for the 360 and so forth.

Real numbers were introduced, somewhat reluctantly on my part, I might add. We finally gave in to many user requests for real numbers. I had taken the position for a long period of time that real numbers were inappropriate in a language such as SNOBOL IV, and were in a sense, antithetical to its structure and concept. But I finally gave in and we implemented real numbers, which of course, we should have had all along. One does have to be able to perform averages, if nothing else, and it's one of those cases where additional feature and complexity that is added to the language to overcome something that was difficult to do, with the facilities that it had. SNOBOL IV grew larger, more sophisticated, making it easier to do things, but the language became more difficult to handle and to learn them, for the programmer.

Right at the end of the year, in 1966, return by name and functions was added again. This is a case where an internal facility, where procedures were returning by name, was suggested in external facilities, an external facility in which a SNOBOL IV procedure could return by name, and therefore introduced, at least conceptually, the defined function analog of a pseudo variable in which the value computed, that a programmer could be a name, a location to which a value could be assigned. It is one of a number of examples where an interna l facility suggested a corresponding external facility, and there is a great deal of parallelism between the internal structure of SNOBOL IV and the external source language itself.

Along at the same time, the so-called recompile facility was enabled at last. Recall that the original version of SNOBOL had conceptually the idea that a label and the name of the string were essentially the same, and that if a statement had a label, then that label corresponded to the name and the statement was its value, the idea being that if you changed the value of the string, which was a label, it should be recompiled in the program and therefore, you could modify your program. As I mentioned earlier, this turned out to be a poor idea because a program could rather inadvertently be destroyed by an operation which unintentionally changed the statement, simply because a variable turned out to be the same as a label. This is particularly bad because it was data dependent with indirect referencing. We could write programs but you had no idea what names would be used, and in which there could be no guarantee given to that, in fact, the program would not destroy itself. Well, SNOBOL I permitted the program to destroy itself but it did not provide for recompilation, only the infamous error message, “Recompile not enabled.” In SNOBOL III we discarded this, having decided it was a bad idea anyway, and not being at all satisfied that simply changing a statement with a label is the way to modify a program or to extend it.

In the beginning of 1967, they said recompile as they had the SNOBOL IV in an entirely different way, again, and that internal structure suggested an external feature. The same internal data objects were used in implementation as they were for source language, and in particular, the source language program was compiled into a block of descriptors of the same nature as was used to represent source language data objects. The idea came to me, at that point, “Well, for heaven’s sake, all we have to do is provide a descriptor pointing to a block of program and we have a way of talking about the program. And therefore, why not use the compiler during execution to convert any string which represents source language program into a block of code, and then permit this to be executed.” This turned out to be trivially easy to do and greatly extended, at least theoretically, the power of the SNOBOL IV language. Originally it was through convert. We had to convert a string and then convert it to code, the compiler was called, and that string was considered as a series of statements separated by semicolons. This corresponding code was compiled and put in a separate block, and a point of that block was returned. The direct go-to permitted the execution to begin with the block, and added this ability to modify or to extend, basically, the program. Eventually the code function was added to simplify the notation, but this is one of the striking examples of the ease in introducing a very complicated feature because of the gene rality and the parallelism between the way that the internal SNOBOL IV system was implemented and the way that the external source language was implemented.

In January of 1967, the allocator was converted to work on the 360. I believe up to that time, we had used an allocator written by Lee Varian; I don’t really remember. Between

March and April of 1966, there was another milestone in the implementation of SNOBOL IV. Stockton Gaines [?] was working at IDA at Princeton, and had been interested in SNOBOL. For a period of time, he became quite interested in the portable machine independent aspects of the implementation. IDA had a 6600, and Stockton Gaines undertook to implement SNOBOL INTERVIEWER going through the same kind of procedure that we had, rewriting the inner system, machine-dependent sub-routines, and writing macro definitions for the outer system macros. Stockton's an incredibly enthusiastic and optimistic person. He as an individual will be described separately. But that was the first implementation of SNOBOL IV on a different machine that was undertaken by someone outside the SNOBOL IV group proper.

The spring of 1967 was devoted to bringing the 360 system up to operating level. We'd gotten a 360-40 at Bell Laboratories, I believe in early '67. (Again, other documentation should be checked for the date.) I remember coming in very, very early in the morning, two, three and four, to assemble a SNOBOL IV system. We had gotten the assembly down to the point by various techniques and understanding of the software to where it only took an hour or so. We also and usually took the machine over every lunch hour and at least once a day, as I recall, assembled the SNOBOL IV system with modifications. This involved a massive amount of computer time. There was no accounting on the machine that was available. The major problem was the survival of the machine hardware through an entire assembly. Over a period of time, of course, it got better, but this was an area of intensive babysitting of the machine. It was my first real opportunity to get hands on the machine, and I was deeply immersed in the machine. It preoccupied me in a large part of my life, as it did Ivan Polansky. It was the first time that he had had an opportunity to get hands on a machine, and the first situation in which he could experiment with it, without being embarrassed or really without putting himself in a position of being inept or put down, Dave Farber always being an expert, being hard to measure up to. Ivan spent a great deal of time on the machine, also.

In June of 1967, the first versions of SNOBOL IV were distributed to selected individuals. These are called preliminary distributions, and there is documentation and various logs as to who these went to and when. These were public distributions of SNOBOL IV, preceding official distribution. They were, of course, bugs. The macro language was changing constantly with the feedback from the 6000 Series of implementation and our own experience with the 360, trying to make it more uniform, trying to document it. The 360 implementations were distributed at this time. For example, I'm sure the University of Michigan got one. Each one differed a little bit, because it was sort of a continuous process. We gave a copy of whatever happened to be current and working at the time.

In July of 1967, the UNIVAC 1108 implementation was started by Lee Osterwilde [?] at the University of Maryland. We now were in a period in which the portability of SNOBOL IV was if not proven, at least quite plausible. There were people who were quite interested, both in the language and the portability aspects, and also they wanted to implement. They began to pick up a number of simultaneous implementations going on in a number of machines. I personally served as consultant on all of these, and did almost all the work in interfacing of the machines. It was quite a practical consideration because the implementations were done by individuals, not corporations. The commitment in terms of manpower and money was minimal, if perhaps the emotional commitment was greater. So beside the 360, the second implementation was begun in July of 1967, on the UNIVAC 1108, not for just some interesting situation, but also in an attempt to simplify the assembly, not knowing what entry points were and what weren't. Lee Osterwilde put stars on all labels in the gain of entry points, completely filled the drum, brought their system down on a horrible thud and crash, and he had to do a dead start. The 1108 implementation required a conversion of the source into quite a different format, because the 1108 assembler expects a somewhat different format than most assemblers. That was done by Osterwilde in the 1108 and I never had any particular contact with it. We did provide syntax tables for them again writing and generating the particular machines codes and internal collating sequence, and generated the so-called machine-independent syntax table.

In August of 1967, the 635 implementation of Murray Hill was started. That was done by Pat Hamilton and Ruth Wyse [?]. It was one of the less successful of the implementations and probably should be discussed separately, as each of these implementations really should be, so as not to confuse among them the various contexts. At this point, distributions were being made outside of Bell Laboratories. The macro language was being constantly revised. Documentation was being developed for the machine-independent aspects of SNOBOL IV. There were three implementation efforts going on outside of our group. Language development also was going on. This also was the summer in which Bernie Dickman and Paul Jenson worked on tracing facilities for SNOBOL IV, again, to be described separately.

In September of 1967, the immediate value assignment was introduced, which changed pattern matching rather noticeably, because up until this time, there was very little that went on during pattern matching that produced side effects during the matching itself. The immediate value assignment introduced that aspect into SNOBOL IV, and was part of the features that eventually led to difficulty with the heuristics in pattern matching of SNOBOL IV, a subject which will be discussed later. The SNOBOL IV system was now undergoing a regular series of edits and a regular format for the source that had been set up. Tracing was completed that summer. In the second half of 1967, a large amount of time was devoted to simply the clerical aspects of revising and constantly revising the macro language, its documentation, correcting bugs, revising the way that things were handled internally. This was largely done by myself, and represented an almost compulsive effort to continual change and approximation approach, hopefully asymptotically to a stable final system. That work is still not done, I might add, as of May, 1972.

Interestingly, beginning of 1968, the implementation book for W.H. Freeman and Company was started. That beginning is now to be culminated by a publication of the book in July of 1972. There will be more about that later. In February of 1968, the idea for the full-scan mode was introduced. Now I alluded earlier to the problem in pattern matching introduced by immediate value assignment, and also I deferred pattern evaluation. That deferred pattern evaluation was generalized in late 1967 to some extent. Nonetheless, the real problem was that now, during pattern matching, side effects were commonplace, at least in complicated patterns. And the heuristics that had been introduced into the algorithm for pattern matching to speed it up now were visible in the sense that parts of the pattern match might not be attempted because of these heuristics, which would prevent certain things from going on that would produce side effects, even though the eventual pattern match might fail. The full scan mode was introduced, simply to turn off all of these and to put the pattern matching in a mode in which there was no attempt made to speed up the operation or to avoid unnecessary attempts at pattern matching, and hence to permit all components of the pattern to be attempted with any possible side effects that these might have.

In March of 1968, we converted SNOBOL IV to the ASP system at Home Bell. ASP stands for Attached Support Processor. It was our replacement for the 7094 and 360 model 50 and 65, as I recall; I don't remember the exact configuration. But it was a conversion for Bell Laboratories at Home Bell, to a 360 machine for the batch environment. The 7094 stayed around for a fair amount of time. SNOBOL IV was one of the first major predecessors to go up on the new 360, which was not surprising, since we had more experience with 360s, having used the Model 40 for over a year, than the systems people who installed the batch processor.

Between March and April of 1968, Version 1 of SNOBOL IV was released. This was a public release, documented, stable release, and represented going public in the official kind of a sense, with some kind of a commitment toward stability and maintenance of individual versions. Documentation was fragmentary up to this point. Immediately after the release of Version I of SNOBOL IV for the first time, we began work on documentation of SNOBOL IV, which eventually to lead to the Prentiss-Hall book, describing the language. We did this using the Text 90 system. The documentation of SNOBOL is itself a separate subject and a lot can be said about machine generation of documents elsewhere.

In May of 1968, external functions were introduced into SNOBOL IV with the help of Howard Strauss, who was on Internship assignment internally in Bell Laboratories. SNOBOL III had external functions. They went under the name of machine language functions. This was an addition to SNOBOL IV to make it possible to extend the language during execution by loading and linking functions written in other languages, most specifically assembly language and FORTRAN to extend the SNOBOL IV system. The period following the release of Version I of SNOBOL IV up through summer again consisted of improvements to SNOBOL IV system, public releases, and work on documentation. In August, CDCs 3600 implementations at Indiana were begun, making this the fourth or fifth independent implementation going on. I don't have directly in front of me the progress of the individual implementation but by this time, certainly the CDC, the 600 implementation was working to some extent, and the portability of SNOBOL IV has really a proven fact. Version II was being worked on at this point, Version II having a number of new language features and minor changes from Version I. Again, these changes are documented in literature.

The negotiations with Prentiss-Hall were underway to publish the SNOBOL IV manual, which was in great demand, which was being distributed free by Bell Laboratories, and the Fall of 1968, was devoted largely to working on the Prentiss-Hall book, which was prepared in now text 360 and printed on our line printers and photo offset by Prentiss-Hall with the line drawings being done by Jim Pogue. Also at this time, the so-called optimization of the SNOBOL IV code was undertaken. The SNOBOL IV so-called outer system being written entirely in macro calls produced very inefficient code. In fact, the nature of the macro language is very much like that of a higher-level language. The operation was variable to variable, with no concepts of registers. This was a considerable help in achieving machine independence, but the resulting output code generated by the macro definition had many obvious redundancies. One macro would load a register, and the next macro from a location in the next macro would load the same register from the same location, and it was obvious that in many cases the instructions could be eliminated. The idea for processing the output of the assembler after macro expansion was mine. Ivan and I worked on it. He assisted me to some extent in writing programs for doing it. This particular aspect developed in the SNOBOL IV again is somewhat peripheral and to be described in more detail later. Basically, the idea was to take the output of the macro expansion and process it in it's machine-readable for in various ways to remove redundant instructions or to improve the sequence of instructions to get faster executing code. This is a major clerical job. It produced a faster running system, and that was used in Version II. The object modules of Version II that were distributed were optimized by a series of programs and some handwork to produce faster running code. The end of 1968 was a terrible scramble, trying to get the book in final form and printed into Prentiss-Hall, saying nothing of negotiations between Bell Laboratories and Prentiss-Hall for the contract of the book. This was finally culminated, I think New Year's Eve or something like that, in 1969, and the book was taken to Prentiss-Hall for publication. As a matter of fact, I think they got the manuscript before the publication agreement was signed.

The period from 1969 on is less well-documented form of diary entries than the preceding part, although there is more existing materials in which to reconstruct that period. It might be worth reflecting this day of the program at this point. In early 1970, the Prentiss-Hall book was published. This represents a significant landmark in some sense, in that it was public documentation in the form of something that was available to everyone. SNOBOL IV had been distributed to a variety of installations to the 360 from our own group in Bell Laboratories, and there were implementations of SNOBOL IV for other machines. Specifically, at the time that the book was sent to the printers right at the end of 1968, in addition to the 360 system, the UNIVAC 1108, the GE 635, the CDQ 6000, the RCA Spectra-70, all had implementations of the Version II of SNOBOL IV running in some fashion or another. SNOBOL IV was thus rather widely available, and was in at this time in fairly stable condition. The bugs and the language were moderately few by comparison. There were fewer bugs in the 360 implementation of SNOBOL IV than there were in most of the IBM support and utility programs.

At that point, in some sense, the SNOBOL IV might have really stopped. However, there was still a lot being done, being that much involved with public distribution, for example, with the significant effort. We were distributing SNOBOL IV ourselves from Bell Laboratories—not only just from Bell Laboratories, but I was practically doing it single-handedly. The reason was several-fold. We strongly felt, or I strongly felt that the only way the distribution could be done in a professional, first-class way was to do it ourselves. A large part of what we were doing in the way of distribution had no particular support from our management. They didn't know we were doing it, probably to the extent that we were. They certainly didn't care. We could have made an attempt to set up a clerical group or something to do it, or could have probably arranged to have it done through program library, but our past experience that indicated that if that was done, it wouldn't be done right, and we felt that to a large extent, the success of the language and the demonstration of professionalism on the part of Bell Laboratories could only be accomplished if we did the work ourselves. For example, I wrote myself all the tapes for distribution of Version II, some 30-odd, as I recall, on one day, and on weekend I had an amount of 40.

So that was occupying us considerably. There was a large amount of correspondence and interaction at this point, many requests for information, people who were interested in implementing SNOBOL IV on other machines, consulting and assistance and providing materials for people who were implementing it or had implemented it on other machines, and so forth. Furthermore, there were obviously things that needed improving, and 1969 became, therefore, a year in which Version III of SNOBOL IV was developed as an incremental improvement over Version II, both in its implementation and in the number of cases in language features.

This year of 1969 produced almost on a same year basis, Version III of SNOBOL IV. I will talk about that more in the next segment, and that was to be the last official released version of SNOBOL IV to the present time, and very likely the last release, period.