Newsletter
Volume 1 Number 2
January 15, 1986

Scratchpad II

IBM Research

In this issue ...

The Scratchpad II Newsletter

An informal quarterly publication of the Computer Algebra Group, Knowledge Systems, Computing Technology Department, IBM Thomas J. Watson Research Center, Box 218 Yorktown Heights, New York 10598.

Editor: Robert S. Sutor

Volume 1, Number 2 January 15, 1986

The Scratchpad II Computer Algebra Group

Manager
Richard D. Jenks
IBM VNET: JENKS at YKTVMZ
CSNET: jenks.yktvmx@ibm
914-945-1233

Algebra
Barry M. Trager
IBM VNET: BMT at YKTVMZ
CSNET: bmt.yktvmz@ibm
914-945-1868

Patrizia Gianni
IBM VNET: GIANNI at YKTVMZ

Compiler
Stephen Watt
IBM VNET: SMWATT at YKTVMZ
CSNET: smwatt.yktvmz@ibm
BITNET: smwatt@yktvmz
914-945-3405

Interface
Rüdiger Gebauer
IBM VNET: GEBAUER at YKTVMZ
914-945-3882

Moss E. Sweedler
IBM VNET: SWEEDLE at YKTVMZ
914-945-2471

Interpreter
Robert S. Sutor
IBM VNET: SUTOR at YKTVMZ
CSNET: sutor.yktvmz@ibm
BITNET: sutor@yktvmz
914-945-2360

Michael Lucks
IBM VNET: LUCKS at YKTVMZ
CSNET: lucks.yktvmz@ibm
BITNET: lucks@yktvmz
914-945-3720

System
Martin L. Brock
IBM VNET: MBROCK at YKTVMZ
914-945-2471

Consultants
David and Gregory Chudnovsky
Professors of Mathematics,
Columbia University.
IBM VNET: SPAD at YKTVMZ

Victor Miller
IBM Research
IBM VNET: VICTOR at YKTVMZ

Cooperative Student
Vladimir A. Grinberg
Columbia University.
IBM VNET: VLAD at YKTVMZ

Secretary
Evelyn Zoernack
IBM VNET: EVELYNZ at YKTVMT
914-945-1187

This Issue

Welcome to the second issue of the ***The Scratchpad II Newsletter***.

This issue contains the first installment of a two-part series about the history of the **Scratchpad** project. Jim Griesmer, the author of the article and the manager of the original group that created **Scratchpad I**, discusses the development of the system and symbolic algebraic computation at IBM Research during the 1965-1977 period.

David and Gregory Chudnovsky, Professors of Mathematics at Columbia University, have contributed their second article to this newsletter series. They describe **Scratchpad II** calculations in algebraic topology and algebraic geometry in "Elliptic Functions and Algebraic Topology," starting on page 2.

Moss Sweedler, visiting the **Scratchpad II** group from the Cornell Department of Mathematics, discusses datatyping in the system in his article "Typing in Scratchpad II." Moss has been with the group since September and describes the abstract datatype system in terms especially appropriate for mathematicians.

Our "Algebra Snapshot" this month describes work done by Dr. Jean Della Dora of the Institut IMAG in Grenoble, France, and Stephen Watt of our group. The article discusses and gives examples of new algebraic facilities for working with differential operators.

The compiler and interpreter sections in "Scratchpad II System Status" have been extended in this issue to give examples of new facilities recently added to **Scratchpad II**. A new section, "Joint Studies and the ALGEBRA 4381", discusses availability of **Scratchpad II**. It lists the current and pending joint study arrangements between the **Scratchpad II** group and the scientific community.

For the convenience of our readers, we have added a section "Some Scratchpad II Constructor Name Abbreviations" on page 8 that lists the names and abbreviations of category, domain and package constructors mentioned in this newsletter.

As I mentioned at the start of the last newsletter, please feel free to contact any **Scratchpad II** group member if you have questions or want information about our system. The names and network addresses of the group appear inside the front cover. To get on our mailing list for future issues, please fill out and return the form on the last page. We're looking forward to hearing from you.

Robert S. Sutor

Typing in Scratchpad II

by
Moss E. Sweedler

This is not about hunt and peck versus touch typing. Rather, here are a few words, for mathematicians, about the **Scratchpad II** datatyping system. If you have wondered about the "Type: ... " message which appears after a computation, this may help. The two main purposes of the datatyping system are to make the behind the scenes context magic possible and to make the compiled **Scratchpad II** code run faster. What I mean by "behind the scenes context magic" is exemplified by *3-5* which could be taking place as integers, integers mod n or as constant polynomials. Depending on how you got to the *3-5*, the system will perform the correct subtraction. Since mathematicians frequently use the same symbol in different contexts without confusion it should be no surprise that the **Scratchpad II** datatyping system is to some extent a model of the organization of mathematics, particularly from a universal algebra point of view. The datatyping system is built around the **Scratchpad II** notions of *category, domain* and *subdomain*.

In a nutshell, the *domain* of an element gives the immediate context for operations on the element such as the "-" in *3-5*. Thus if *3* and *5* are the *3* and *5* in the domain **Integer**, then "-" is the "-" of **Integer** rather than the minus of the integers mod n. The domain of an element can be thought of as the main set in which the element lies. The term "main" is used because *3* and *5* as elements of **Integer** also lie in the subsets **NonNegativeInteger**, **PositiveInteger** and **OddInteger** which are also named sets in **Scratchpad II**. **NonNegativeInteger**, **PositiveInteger** and **OddInteger** are useful for many purposes but they are not closed under subtraction. **Scratchpad II** works with one subtraction operation which is defined on the set **Integer** rather than having each of **NonNegativeInteger**, **PositiveInteger** and **OddInteger** equipped with a subtraction operation which gives an **Integer**. If **Integer** is the domain in which *3* lies, **NonNegativeInteger**, **PositiveInteger** and **OddInteger** are all *subdomains* containing *3*. In **Scratchpad II** a subdomain is simply a subset of a domain determined by a predicate (that is, a boolean-valued condition) on that domain. In terms of *datatypes* the domain of an element could be thought of as the element's (main) type and the various subdomains the element lies in are its subtypes. The result of each computation lies in a domain or subdomain, which is the "Type" which appears on the screen beneath the computation. An

element can lie in many subdomains but always lies in a unique domain. Since operations are associated with domains, it is by knowing the unique domain of an element that the system knows which operation to use. Conflicts may arise. Suppose *3* is the **Integer** *3* and *5* is a constant polynomial *5*. In this case the system finds in its database that the **Integer** *3* may be mapped, *coerced* is the **Scratchpad II** term, to the constant polynomial *3*. Then the *3-5* is performed as the difference of constant polynomials and that is the resulting type. Talking about the **Scratchpad II** database leads to the notion of *category*.

The **Scratchpad II** meaning of the term *category* is not the same as the mathematical meaning of the term, although it is what a student usually thinks of when first encountering categories. The **Scratchpad II** meaning of category is a class of objects with similar operations and attributes. For example, the category **SemiGroup** consists of **Set** with a product operation and the attribute that the product operation be associative. Morphisms are not part of the definition of category. In **Scratchpad II**, categories exist solely for database purposes. For example, creating a specific domain which is a **SemiGroup** requires writing executable code for the product operation. Creating a category such as **SemiGroup** only requires specifying for the database what the operations and attributes are. Previous entries in the database may be built upon. Thus once the database knows what **Set** is, **SemiGroup** may be specified as **Set** with an associative product operation. By keeping track of the category in which a domain lies, the system knows what operations are available for elements of the domain. Moreover, the operations may involve elements other than those of the domain in the category. For example if R is a **Ring**, i.e. in the category **Ring**, then there is a category **Module(R)** which is the category of left R modules. For M in **Module(R)** there is the operation

$$"*" : (R, M) \to M$$

which is simply the module action of R on M.

The category of R modules, **Module(R)**, exhibits the fact that categories may take parameters. In this respect **Scratchpad II** categories are like usual mathematical categories. Categories may *extend* other categories as illustrated by the category **SemiGroup** extending the category **Set**. The **Scratchpad II** terminology is to say that **SemiGroup** is a *join* of **Set** with the product operation and the associativity attribute. The system keeps track of when a category is specified to be a previously defined category joined with additional operations or attributes or both. Thus it knows that a **SemiGroup** becomes a **Set** when the product operation is dropped. As a result the **Scratchpad II** categories have "forgetful functors" with pretty much the usual mathematical meaning.

Elliptic Functions and Algebraic Topology

by
David and Gregory Chudnovsky

In this note we discuss some recent work of ours, where **Scratchpad II** computer algebra calculations helped to answer a problem of Landweber and Stong concerning the description of characteristic classes of Spin-manifolds with a nontrivial action on S^1.

We recall some standard definitions from [1]. Let $p_0 = 1$ and $p_1, p_2, \dots$ be indeterminates. A product $p_{j_1} \cdots p_{j_r}$ in $p = (p_0, p_1, p_2, \dots)$ has weight $j = j_1 + \cdots + j_r$. A multiplicative sequence $\{K_j(p)\}$ of polynomials in p ($K_0 = 1$) is a sequence such that any power series identity

$$\sum_{i=0}^{\infty} p_i z^i = \left(\sum_{i=0}^{\infty} p'_i z^i\right)\left(\sum_{i=0}^{\infty} p''_i z^i\right)$$

implies the identity

$$\sum_{i=0}^{\infty} K_j(p) z^j = \left(\sum_{i=0}^{\infty} K_j(p') z^j\right) \left(\sum_{i=0}^{\infty} K_j(p'') z^j\right).$$

A multiplicative sequence is completely determined by its characteristic series

$$P_K(x) = \sum_{n=0}^{\infty} K_n(x, 0, \dots, 0).$$

For a manifold M^{4k}, a genus $\phi([M^{4n}])$ associated with $P_K(x)$ is

$$\phi([M^{4n}]) = K_n(p_1, \dots, p_n)[M^{4n}],$$

where $p_i = p_i(M) \in H^{4i}(M)$ is the i-the Pontryagin class of a tangent bundle of M. So, for example, the *signature*, $\tau(M)$ of M, corresponds to the characteristic series $x/\tanh(x)$.

Let M be a Spin-manifold with a nontrivial action of S^1. This action is *odd* if $M^{S^1} \neq 0$ and for $\mathbb{Z}_2 = \{\pm 1\} \subset S^1$, the components of $M^{\mathbb{Z}_2}$ have codimensions $\equiv 2 \pmod 4$. According to [2] and [3], a signature vanishes on a (closed and connected) Spin-manifold M with a nontrivial action of

S^1. Landweber and Stong started a project to describe all characteristic classes vanishing on all such manifolds [3.]. For this one has to describe characteristic classes vanishing on $[\mathbb{CP}(\xi^{2m})]$, $\xi^{2m} \to B$ being a complex vector bundle of even dimension over a (compact) oriented manifold. Landweber and Stong posed to us the problem of proving the integrality of (rational) coefficients of a universal genera, vanishing on all $[\mathbb{CP}(\xi^{2m})]$. In their analysis [3], this universal genera $f_t(y)$ was determined through a certain nonlinear differential equation. Namely, ([3, Ch. 5]), one puts

$$t f_t(y) = m(y t^2,\ y) - y$$

for $m(v, t) \in \mathbb{Q}[[v,\ t]]$, $m = v + v^2(\ \dots\ )$. The series m is completely characterized by the following property:

$$(i) \qquad \begin{aligned} m &= \frac{v}{1-v} + t\,\phi_1 + t^2\,\phi_2 + \cdots \\ &= v + v^2\,\psi_2 + v^3\,\psi_3 + \cdots \end{aligned}$$
$$\text{where } \phi_i \in v^2\,\mathbb{Q}[[v]],\ \psi_i \in \mathbb{Q}[[t]]$$

and

$$(ii) \qquad \begin{aligned} 0 \equiv\ & 2v^2(1+4t^2)m_{vv} - 4t^2 v m_v + 12t^2 v \\ & + (12v - 8t)tm - 12tm^2 - 4m^3 \end{aligned}$$

Using **Scratchpad II** we started to look at coefficients of $f_t(y)$. Coefficients for the first hundred terms were integral and grew quickly. (This immediately suggested a possible relationship with modular forms and θ-functions.) We reduced the order of the differential equation, identified its solutions with properly normalized elliptic integrals of the first kind, and, having enough terms of the power series expansions, found a unique expression of $f_t(y)$.

At the end, we found an expression for $f_t(y)$ as a ratio of θ-functions. For this one put

$$t = -\sum_{n=1}^{\infty} \frac{(2n+1)q^{2n+1}}{(1-q^{2n+1})}.$$

(So that $t = -q + O(q^2)$.) Then

$$f_t(y) = \prod_{n=1}^{\infty} \frac{(1-yq^{2n-1})/(1-q^{2n-1})^2}{(1-yq^{2n})/(1-q^{2n})^2}.$$

Note that in classical Jacobi notation,

$$t = \frac{1}{24}\left\{1 - \Theta_2(0)^4 - \Theta_3(0)^4\right\}.$$

Incidentally, all coefficients of $f_t(y)$ are integral.

References

[1] Hirzebruch, F.. *Topological Methods in Algebraic Geometry*, 1956, Grundlehren der math. Wissenschaften, Vol. 131, Third enlarged edition, Springer-Verlag, Heidelberg, 1978.

[2] Atiyah, M.F., and Hirzebruch, F.. *Essays on Topology and Related Topics*, 1970, 18-28.

[3] Landweber, P. S., and Stong, R.E.. *Topology* (to appear).

A History of the SCRATCHPAD Project (1965-1977)

by
James H. Griesmer

In the spring of 1965, Ralph Gomory approached me and asked me to begin a project whose goal would be to develop a computer system suitable for a mathematician to use in trying out ideas and conjectures. I recall one of the things that influenced Ralph was his seeing a RAND tablet and imagining that device being used for the input of mathematical expressions. The project was called Special Projects (later changed to Symbol Manipulation) and, for a while, consisted only of myself. The project was to be a relatively small one, with our relying on the efforts of others, as well as our own.

I spent the first year becoming familiar with the field of what was then called symbolic and algebraic manipulation, and is now called symbolic algebraic computation, or computer algebra. During the summer of 1965 I visited the IBM Boston Programming Center to participate in a workshop on 7090/94 **FORMAC**.

The ACM Special Interest Committee on Symbolic and Algebraic Manipulation (SICSAM, and later SIGSAM) was organized by Jean Sammet during 1965 and she served as its first chairperson. Without question, SICSAM/SIGSAM has been a major force in the development and promotion of this field since that date. The first major technical event sponsored by this organization was the ACM Symposium on Symbolic and Algebraic Manipulation, chaired by Jean, and held in Washington in March, 1966. (I served as Publicity Chairman.) The systems given major prominence at that conference were 7090/94 **FORMAC, ALPAK** and **Formula**

ALGOL. Papers were also given on **REDUCE** (though not called that yet) by Tony Hearn and the Korsvold simplifier. The major result announced at that conference was the work on computing polynomial GCD's by George Collins. George was in our department at the time, but was about to accept an appointment at the University of Wisconsin.

The other organization which was notable in promoting the field was a SHARE project consisting primarily of 7090/94 **FORMAC** users (and later **PL/I-FORMAC** users). It was headed by Betty Cuthill and later by Bob Tobey.

During late 1965 I joined forces with Fred Blair, who was the local LISP expert and maintained **LISP 1.5** on both the 7044 and 7094. This early period were spent obtaining and understanding symbolic computation programs written in LISP. In late 1965 we were visited by Knut Korsvold and received a copy of his simplifier which we put under 7094 LISP. We received a considerable lesson in intermediate expression swell when we tried to (and eventually did) solve one of a set of problems posed by Max Goldstein using the Korsvold simplifier. Other key acquisitions in that period were **REDUCE** (rational function simplifier and command interpreter) from Tony Hearn then at Stanford, **MATHLAB** (rational integration, factoring, Laplace transform) from the late Carl Engelman at Mitre, **CHARYBDIS** (two-dimensional output) from Jonathan Millen at Mitre, and **SIN** (symbolic integration) from Joel Moses at MIT. Joel came to our laboratory to give a seminar on **SIN** with a box of punched cards under his arm! Tony Hearn visited our lab at least two times in that period. The linear equations solving work of John Lipson was also converted into LISP.

Toward the end of 1966, the late Bill Martin joined us as a consultant for one year and helped us bring his Picture Compiler into our environment. Joe Harry from the Computing Center wrote a program for the IBM 1130/2250 system to display algebraic expressions using output from the Picture Compiler.

With this large battery of algebraic facilities written in LISP, it was clear that we would have to develop a new LISP system to supersede **LISP 1.5** on the 7094. We also wanted to move to an interactive environment, which had been an objective from the start. Fred Blair (with some advice from Stu Toledano) designed a **LISP 1.5** for S/360 which gave us double (and eventually, with further work, even more) address space, while allowing this existing battery of programs to run with a minimum of conversion. The basic addressing mode devised by Fred used half-word (16 bit) pointers to full words, which allowed a total of 256K bytes in the space managed by the garbage collector. Fred, Joe Harry and I worked on **S/360 LISP** for about 15 months (January 1967-Spring 1968) before some substantial applications began to run. Lou Hodes, then at the National Institutes of Health, spent a few months with us in 1967 and helped with this work. The implementation method was to bootstrap using 7094 LISP to generate S/360 code and structures. Because the bootstrapping process was so cumbersome, involving the 7094, 7044 and the S/360, we resorted to such quick fixes as patch cards in hexadecimal to help speed up our debugging work. Eventually the bootstrapping activity was done on **S/360 LISP** itself, which represented a substantial workout for **S/360 LISP**. Fred received an IBM Research Outstanding Contribution Award for his design work on **S/360 LISP**.

The first implementation of **S/360 LISP** was for a batch environment under OS/360. This was followed almost immediately by a version running interactively under TSS/360 using a 2741 terminal for user interaction. Then Joe Harry and I, together with Joan Jaffe of the Computing Center and Knut Bahr (who was visiting us), wrote the telecommunications facilities for connecting the 1130/2250 with TSS, which enabled us to see the results of algebraic computations as prepared by the Picture Compiler or by **CHARYBDIS** on a display terminal. (Knut spent the majority of his stay at our laboratory converting **PL/I-FORMAC** to run on the IBM S/360 Model 91.) The integration of the algebraic facilities with the output facilities on TSS was accomplished by mid-1969. It was also clear that a multiple precision integer package was needed, and I worked for some time during 1969 and 1970 on these routines. Knut Bahr described the **FORMAC** representation for "bignums", and this representation was carried over to **S/360 LISP** using the vector data-type. The algorithms which were used were those in Knuth [11].

Fred Gustavson from our department wrote a set of S/360 routines which implemented the hash-coding simplification ideas of Bill Martin. In the end, this approach did not seem to be competitive with the algorithms for rational function simplification present in **REDUCE**.

In 1968 our second effort to bring Dick Jenks to our group was successful. Dick set to work defining an input language for mathematicians, an effort that culminated in the user language of the **Scratchpad** I system. Bob Risch also joined our department in 1968 and continued his work on algorithms for symbolic integration.

By 1970 we were convinced that the system which had evolved needed a name. Jean Sammet, in a review of symbol manipulation systems in the spring of that year, referred to the system as a "conglomerate system." A half-day session involving Fred, Dick and myself produced the name **Scratchpad**. (IBM lawyers put the name through the trademark system to ensure that it was not in use by a competitive company or product. Actually, in those days, the greatest confusion was between **Scratchpad** and SKETCHPAD, the pioneering graphics system of Ivan Sutherland.) Unfortunately, the name was not chosen in time for the first paper on our work presented in Bonn in 1970 [1]. This paper described the amalgamated set of facilities, which included the work of Hearn, Engelman, Moses, Korsvold and Lipson, accessed by a top level language that was simply an extension of that used in **REDUCE**. Communication between different facilities was done by passing expressions in prefix form. The paper also described the user language that Dick had developed, the full use of which would come in the version of the system that was demonstrated a year later.

While in Europe, I visited the Cambridge group (David Barton, Steve Bourne, John Fitch and John Horton), Yvon Siret at Grenoble, and Max Engeli at Zurich. At this time, I knew that I would be involved with Carl Engelman in developing sessions on symbolic systems at the forthcoming Second Symposium on Symbolic and Algebraic Manipulation (SYMSAM/II), and thus I wanted to see first hand as much of the work on systems as I could. Earlier, I had also visited Applied Data Research in Wakefield, Massachusetts, to see the **IAM** work under Carlos Christensen.

In the spring of 1970, the system received a considerable shakedown when it was exhibited at an IBM Open House for families of employees. Some of the young people were fairly sophisticated mathematically and gave the system a good workout, including some integration problems.

In the fall of 1970, I had arranged to go on sabbatical at the University of California at Berkeley for the 1970-71 academic year. The work on **Scratchpad** and the design of the new version was carried on by Dick, who had a 2741 terminal installed in his home. By this time **S/360 LISP** had been brought over to CP/CMS, and this became the development environment, still using the 2741 terminal as the user interface. (Over the years **S/360 LISP** on CP/CMS was given considerable enhancement by Mark Pivovonsky.) It was this version of **Scratchpad**, with its user language, that was presented and demonstrated at SYMSAM/II in March 1971 [2]. While the large set of collected algebraic facilities remained in the system, they were now accessed through the user language and a "Markov algorithm" evaluator. The user language translator was created using **META/LISP**, an interactive translator writing system. At SYMSAM/II, the other general-purpose and polynomial systems described were **Macsyma, MATHLAB, PL/I-FORMAC, IAM, REDUCE2**, the **Cambridge Algebra System, SAC-1** and **ALTRAN**.

Later that summer, **Scratchpad** was also demonstrated at the SHARE Meeting in New York City [4]. Dick was given an IBM Research Outstanding Contribution Award in 1972 for his design of the **Scratchpad** system and language. (In a paper that Jean Sammet wrote for the 25th anniversary issue of the *IBM Journal of Research and Development* in 1981, entitled "History of IBM's Technical Contributions to High Level Programming Languages," she included a section on the **Scratchpad** language. As significant contributions, the article listed four major high-level languages: **APL, FORTRAN, GPSS, PL/I**, and five additional languages: **Commercial Translator, FORMAC, Scratchpad, QUIKTRAN** and **CPS**.)

Dick gave papers on the **META/PLUS** syntax extension facility at IFIP '71 in Ljubljana, Yugoslavia, and on the two-dimensional input language at a 1972 SIGPLAN conference at Los Alamos [6].

Our efforts turned to developing a user community at the Research Center. In a three-week period in January and February, 1972, eight one-day courses were given to students seated at their own 2741 terminals. Each day consisted of a set of talks by Dick and me, with intervening time for the students to try out the facilities that had just been presented. The teacher's terminal station consisted of a 2741 with a television camera attached to pick up the typing activity and display it on a TV screen. Needless to say, the lab sessions were fairly noisy.

One of our big problems with **Scratchpad** was the large collection of binary programs that comprised the system. We had relied on a mechanism developed for **S/360 LISP** which migrated programs to secondary storage to make way for needed programs or additional storage for list structures (our own version of "paging"). This led to a good deal of "thrashing" and performance suffered. Our solution to this was to load binary programs outside of LISP's address space, and that provided considerable relief.

Some new features incorporated in **Scratchpad** in 1972 were the history file (which allowed the user to backtrack), system commands, and **APL** features. A description of this new version of the system was reported at ONLINE 72 at Uxbridge in the United Kingdom [5]. Following the ONLINE conference, the system was demonstrated at Julich, West Germany using a computer in West Berlin. Emphasis was placed on stabilizing the system, and, except for the addition of new capabilities, this version became the foundation for **Scratchpad** development and use in applications for the next several years.

Semi-annual courses, demonstrations for visitors and working with users continued. Two visitors in 1972-73, Hale Trotter from Princeton and W. Kahan from Berkeley, were very helpful with their critical comments. A folding user reference card was published in 1974, and, after several preliminary versions appeared, a more polished User's Manual was published in 1975 [8].

David Yun joined our group in 1973. David continued his research work in algebraic algorithms, but at the same time promoted the use of **Scratchpad** on some very key applications. I remember one that involved the use of **Scratchpad** on applications in the area of discrete Fourier transforms, and a second, with Will Miranker of our department, involved the finite element method. The first application required the addition of a number of new **Scratchpad** functions, such as ones for solving a Diophantine equation and inverting one polynomial modulo a second.

David influenced Paul Wang to come to our laboratory for two consecutive summers, and worked with Paul in bringing Paul's factorization work to **Scratchpad**. Rudiger Loos also made several lengthy visits to our group, and provided extensive capabilities for computing with polynomials over finite fields. He, like the other visitors mentioned above, provided considerable stimulation to our project in the form of new ideas and critical evaluation of what we were doing.

It was about this time that 3277 terminals began to show up in our offices. This provided a considerable improvement in productivity as compared with the 2741.

In order to fit in the large number of LISP programs that now existed with the addition of Paul Wang's factorization work, **Scratchpad** was made available in two versions, one with a full set of algebraic capabilities, but leaving out symbolic integration (SIN), and the second version with a reduced set of algebraic capabilities and with the symbolic integration facilities included.

Fred Blair was influenced by this need to have a larger address space, and by developments in the LISP community, to argue for a new LISP development effort. This work gave rise to **LISP/370**, which later led to **LISP/VM**.

In 1973-74 we were fortunate to have Arthur Norman spend a year with us as a postdoctoral fellow. In addition to his own research activities, Arthur involved himself in the **Scratchpad** effort and implemented a very extensive power series package.

During this time Dick served as Editor of the SIGSAM Bulletin (1970-1975), and I served first as Vice-Chairman (1971-73) and then as Chairman of SIGSAM (1973-75).

Dick's ideas on creating a "mode-based" version of **Scratchpad** were articulated in a paper given at a SIGPLAN conference in 1974. This could be said to be a first explication of Dick's thinking about languages and systems for computer algebra which culminated in **Scratchpad II**.

A European meeting, EUROSAM '74, took place under the chairmanship of Hans van Hulzen with Dick acting as Proceedings Editor. The Systems Sessions at EUROSAM included papers or demonstrations on **CAMAL, FORMAC, ANALYTIK, REDUCE, SCHOONSHIP, MATHLAB, Scratchpad** and **Macsyma**.

We also began to preach the **Scratchpad** gospel to other IBM locations, both through individual talks and demonstrations, and through courses. I gave a course at IBM Burlington early in 1975, at the IBM Research Laboratory at San Jose in May, and talks and demonstrations to three separate groups at IBM Rochester over a day and a half in October. Just prior to my visit to Rochester, Dick and I demonstrated **Scratchpad** at ACM 75 in Minneapolis, joining Joel Moses, Carl Engelman and Tony Hearn who were did the same with their respective systems: **Macsyma, MATHLAB** and **REDUCE.**

A note on the FORMAT statement was published in 1975 [10]. This facility, which provided a convenient notation for a user to specify the formatting of expressions for output, was probably the final major addition to **Scratchpad**.

Keeping to our schedule of holding a symposium here in the United States every five years, the 1976 ACM Symposium on Symbolic and Algebraic Computation took place at our laboratory in August. I was the General Chairman, Bob Caviness was the Program Chairman, and Dick was the Proceedings Editor. In a sense, my efforts as General Chairman probably represented my "swan song" relative to

my active involvement in the field. By the time the conference was held, I had begun to serve as Manager of Education here at the Research Center, a position I held until 1981. Although I continued my strong interest in the field, my active participation was limited to serving as an Associate Editor (representing SIGSAM) for the *ACM Transactions on Mathematical Software*.

Dick left on sabbatical to the University of Utah during 1976-77 (followed by a summer at the IBM Palo Alto Science Center). Upon his return he began implementation work on **Scratchpad II**, with ideas he had been developing for some time. But that is the next part of the **Scratchpad** story.

In reflecting on this 12 year period, I would say the major shift in the field of symbolic and algebraic computation that took place was the increasing emphasis placed on algebraic algorithms. Initially, the work on systems and applications had primacy. However, as the field matured, questions of improved algorithms became of equal importance. Perhaps the best way to see this shift is to compare the programs for the 1966 and 1971 symposia, with that for the 1976 conference.

In conclusion, I would like to use this opportunity to express my thanks to the individuals mentioned above who made my years of working in the field of computer algebra so gratifying. In addition, I wish to acknowledge the contributions of the two people who played key roles in the success of **Scratchpad**: Dick Jenks, for the language and system design, and Fred Blair, for the design of **S/360 LISP**.

I would also like to thank two other sets of individuals whose help and support was so important to my work and to our efforts in this field. During my association with the project, it enjoyed support from the following directors of the Mathematical Sciences Department: Ralph Gomory, Hirsh Cohen, Sam Winograd, Alan Hoffman and Dick Toupin. Finally, there were many individuals in the Computing Center who provided consultation and support, often in ways which were beyond their normal responsibilities. The following individuals come to mind: the late Emil Cohen, Bob McNeill, Frank Costa, Norm Pass, Al Weis, Dave Mainey and Carol Thompson.

References

[1] Blair, F. W.; Griesmer, J.H.; and Jenks, R. D.. "An Interactive Facility for Symbolic Mathematics," *Proceedings of the International Computing Symposium '70.* Bonn, Germany: Gesellschaft fur Mathematik und Datenverabeitung, May 1970, pp. 394-419. (Also IBM Research Report RC 2766, January 21, 1970).

[2] Griesmer, J.H., and Jenks, R. D.. "**Scratchpad**/1 - An Interactive Facility for Symbolic Mathematics,", *Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation*, Edited by S. R. Petrick. New York: Association for Computing Machinery, March 1971, pp. 42-58. (Also IBM Research Report RC 3260, February 23, 1971.)

[3] "**Scratchpad**/1 - An Interactive Facility for Symbolic Mathematics", U.S. Army Research Station, Durham, NC. *Proceedings of the 1971 Army Numerical Analysis Conference*, Washington, DC, April 1971, pp. 1-10.

[4] "**Scratchpad**/1 - An Interactive Facility for Symbolic Mathematics," *Proceedings of SHARE XXXVII*, New York, NY, August 1971.

[5] Griesmer, J.H., and Jenks, R. D.. "The **Scratchpad** System", *Proceedings of the ON-LINE 72 Conference.* Brunel University, Uxbridge, Middlesex, England, September 1972. (Also IBM Research Report RC 3925, July 12, 1972.)

[6] Griesmer, J.H., and Jenks, R. D.. "**Scratchpad**: A Capsule View," *SIGPLAN Notices: Proceedings of Two-Dimensional Man-Machine Communication Symposium.* Edited by M. Wells. New York: Association for Computing Machinery, Vol. 7, No. 10 (October 1972), pp. 93-102. Symposium held at Los Alamos, New Mexico. (Also IBM Research Report RC 3972, August 1972).

[7] Griesmer, J.H., and Jenks, R. D.. "The **Scratchpad** System," *Proceedings of the SHARE European Association Meeting*, SEAS 73, Leuven, Belgium, September 10-14, 1973.

[8] Griesmer, J.H.; Jenks, R. D.; and Yun, D. Y. Y.. *The* **Scratchpad** *User's Manual.* IBM Research Report RA 70, June 1975, Yorktown Heights, NY.

[9] Griesmer, J.H.; Jenks, R. D.; and Yun, D. Y. Y.. "A **Scratchpad** Solution to Problem No. 7," *SIGSAM Bulletin*, New York: Association

for Computing Machinery, Vol. 9, No. 3 (August 1975), pp. 13-17.

[10] Griesmer, J.H.; Jenks, R. D.; and Yun, D. Y. Y.. "A Format Statement in **Scratchpad**," *SIGSAM Bulletin*, New York: Association for Computing Machinery, Vol. 9, No. 3 (August 1975), pp. 24-25.

[11] Knuth, Donald E. *The Art of Computer Programming.* Vol. 2: *Seminumerical Algorithms.* 2nd ed. Reading, Massachusetts: Addison-Wesley Publishing Company, 1981.

Some Scratchpad II Constructor Name Abbreviations

The following are some of the abbreviations used for category, domain and package constructors in this newsletter.

Abbreviation	*Constructor Name*
A	Any
B	Boolean
E	Expression
DPMM	DirectProductMatrixModule
EF	ElementaryFunction
FSET	FiniteSet
G	Gaussian
GF	GaloisField
I	Integer
KAF	KeyedAccessFile
L	List
LODO	LinearOrdinaryDifferential-Operator
M	Matrix
NNI	NonNegativeInteger
ODR	OrdinaryDifferentialRing
P	Polynomial
Q	Quaternion
QF	QuotientField
RN	RationalNumber
S	String
SE	SortedExpressrions
SMDR	SquareMatrixDifferentialRing
TBL	Table
UPDR	UnivariatePolynomial-DifferentialRing
V	Vector

Scratchpad II System Status

This section is used each issue to discuss the state of the components of the developing **Scratchpad II** system. The full names of constructors abbreviated in the following are contained in "Some Scratchpad II Constructor Name Abbreviations".

The Compiler

One of the side effects of writing the new compiler in **Scratchpad II** has been that a collection of generally useful data structures has been added to the **Scratchpad II** library. The new domains include a number of table types (keyed access files, hash tables and association lists), finite sets, a general tree type, stacks and tiered tables, and a list type which supports an efficient, sharing cross product operation. This column will describe hash tables, keyed access files and finite sets.

Hash Tables

The hash table domain constructor takes two arguments: a domain of keys and a domain of entries. The following assignment creates a hash table in which polynomials can be stored, indexed by rational numbers.

```
t: TBL(RN, P I) := table()

   (1)  table()

Type: TBL(RN,P I)
```

Entries can be saved in the table in the following way:

```
t 2 := x**2 + y

                 2
   (2)  y + x

Type: P I

t (1/3) := z**3 + y*z

          3
   (3)  z  + z*y

Type: P I

t

                 1    3                  2
   (4)  table(---= z  + y*z,2= y + x  )
                 3

Type: TBL(RN,P I)
```

There are two ways to find the value associated with a particular key. Under most circumstances, it is known in advance that the desired entry exists and

that it would be an error otherwise. In this case application is used to obtain the entry.

```
t 2

                2
   (5)  y + x
Type: P I

t 3

================================
Error detected within algebra code:
   No such element in table
================================
```

Often it is necessary to search a table, returning the entry if there is one and a failure result if there isn't.

```
search(t, 2)

                2
   (6)  y + x
Type: Union(P I,failed)

search(t, 3)

   (7)  "failed"
Type: Union(P I,failed)
```

The # function tells how many entries are in the table and the ***keys*** function returns a list of the indices to the table.

```
# t

   (8)  2
Type: NNI

keys t

          1
   (9)  [---,2]
          3
Type: L RN
```

The ***remove*** function searches the table for an entry and removes it if one is found.

```
remove(t, 1/3)

          3
   (10)  z  + z*y
Type: Union(P I,failed)

t

                  2
   (11)  table(2= y + x )
Type: TBL(RN,P I)
```

Keyed Access Files

The following assignments create a temporary file and then save a polynomial in it.

```
f: KAF P I := table()
   (12)  [KAF0,LISPLIB,*]
Type: KAF P I

f "two" := x**2 + y

                2
   (13)  y + x
Type: P I
```

The domain **KAF P I** belongs to the category of tables with strings as keys and polynomials over the integers as entries *and* to the category of files of polynomials over the integers. Since **KAF P I** belongs to the table category, all of the operations described for hash tables can be used. Additionally, operations for opening files by name and for closing them are available.

```
-- OLDPOLYS is a previously created file
g: KAF P I := open "OLDPOLYS"
   (14)  [OLDPOLYS,LISPLIB,*]
Type: KAF P I

-- There are three entries in the file.
-- The keys to the entries are listed by ...
keys g
   (15)  ["badexample","goodexample","bestexample"]
Type: L S

g "bestexample"

          2
   (16)  x  - x + 1
Type: P I

g "bestexample" :=  % - 2*x

          2
   (17)  x  - 3x + 1
Type: P I

close g
   (18)  [OLDPOLYS,LISPLIB,*]
Type: KAF P I

close f
   (19)  [KAF0,LISPLIB,*]
Type: KAF P I
```

By using keyed access files, users can save objects from session to session, thereby extending the existing workspace history facility in **Scratchpad II**.

Finite Sets

Finite sets provide an aggregate data structure which supports a number of set operations: union, intersection, set difference, symmetric difference,

complement (for finite domain), and membership and subset testing. Unlike lists, finite sets are unordered and do not contain duplicates.

A finite set is created using braces:

```
s1 := {1,2,4,6..9,11}
   (20)  {1,2,4,6,7,8,9,11}
Type: FSET I

s2 := union({1..7}, {5..11})
   (21)  {1,2,3,4,5,6,7,8,9,10,11}
Type: FSET I

intersect(s2, {3,4})
   (22)  {3,4}
Type: FSET I

7 in s1   -- membership test
   (23)  true
Type: B

s1 <= s2  -- subset test
   (24)  true
Type: B
```

Sets with members from a finite domain have a ***complement*** operation

```
s3: FSET GF 5 := {1,2,3}
   (25)  {1,2,3}
Type: FSET GF 5

complement s3
   (26)  {4,0}
Type: FSET GF 5
```

Stephen M. Watt

The Interpreter

Generally speaking, the performance problems noted in the last issue have been resolved. We are still working on improving modemap selection (finding an applicable function with given arguments, possibly by changing the datatypes of the arguments) and general domain tower coercion (inverting, say, an element of **Gaussian Polynomial RationalNumber** to get an element of **RationalFunction Gaussian Integer**). The rest of this section will discuss improvements to the interpreter in the last few months.

Interface

We are now in the process of improving the error messages produced by the interpreter. Long-time users will note that the messages are now somewhat more verbose, identifying the error and (we hope) helping the users to better understand the capabilities of **Scratchpad II**.

Scratchpad II now allows users to choose one or both of two computational output forms: 2-dimensional algebraic format (default) or **FORTRAN** format. format. To turn on **FORTRAN** formatting, issue

```
)set output fortran yes
```

Specify *no* instead of *yes* to turn it off. A single **FORTRAN** line will be displayed on one line. For medium-sized expressions (20 **FORTRAN** lines or less), continuation characters are used as necessary. If possible, large expressions (more than 20 **FORTRAN** lines) are broken into summation terms of 20 or fewer lines.

```
p := (x-y)**10

FORTRAN expression is:

      P=((y**10)+((-10*x)*(y**9))+((45*(x**2))*(
     Cy**8))+((-120*(x**3))*(y**7))+((210*(x**4)
     C)*(y**6))+((-252*(x**5))*(y**5))+((210*(x*
     C*6))*(y**4))+((-120*(x**7))*(y**3))+((45*(
     Cx**8))*(y**2))+((-10*(x**9))*y)+(x**10))


   (1)
       10        9      2 8       3 7       4 6
      y   - 10x*y  + 45x y  - 120x y  + 210x y
   +
            5 5       6 4       7 3      8 2
      - 252x y  + 210x y  - 120x y  + 45x y
   +
           9     10
      - 10x y + x
                                                  Type: P I
```

To suppress all output from a statement, end the line with a semicolon. It will soon be possible to redirect the regular and **FORTRAN** output to a file.

Though **Scratchpad II** system developers have long had the ability to trace the execution of the functions underlying **Scratchpad II**, tracing of compiled user functions has only recently been implemented. When used with the *)math* option, the *)trace* command allows one to see the input and output to user maps printed in an algebraic format.

```
-- the function largeDen will exit once the
-- argument denominator to it is larger than 10000

largeDen q ==
  denom q > 10000 => q
  largeDen ((numer q) / (1 + (denom q)**2))
```

```
-- on first use, the function will be compiled

largeDen (13/29)
   compiling largeDen with signature RN -> RN

                13
   (2)  --------
                708965
                                                    Type: RN


-- Once compiled, the function can be traced.

)tr largeDen )math
    largeDen traced

-- Let's compute the result again, viewing
-- intermediate results.

largeDen (13/29)

1 <enter largeDen :
  13
 ----
  29
 2 <enter largeDen :
   13
  -----
   842
  3 <enter largeDen :
     13
    --------
    708965
  3 >exit  largeDen :
     13
    --------
    708965
 2 >exit  largeDen :
   13
  --------
  708965
1 >exit  largeDen :
   13
 --------
 708965

                13
   (3)  --------
                708965
                                                    Type: RN
```

Using !

The operator *!* allows one to map functions over the entries of such "aggregate" domains as **List, Vector, FiniteSet, HashSet, Matrix, SquareMatrix** and **RectangularMatrix**. In the present implementation,

```
f ! a
```

is translated into

```
reshape([f e for e in ravel a],a)
```

ravel is a function that takes an aggregate domain object and creates a new object which is a list of the elements of the aggregate. The ***reshape*** function is paired with ***ravel*** and knows how to rebuild the aggregate form from the list form. ***ravel*** and ***reshape*** are presently defined in packages for the domains above.

The function ***parity*** returns 1 for odd integers, 0 for even ones.

```
parity x ==
  oddp x => 1
  0
```

parity can be applied to each entry in a matrix by using *!*.

```
m : M I := [[1,2,3],[4,5,6]]

       +1  2  3+
   (2) |       |
       +4  5  6+
                                                    Type: M I


parity ! m
   compiling parity with signature I -> I

       +1  0  1+
   (3) |       |
       +0  1  0+
                                                    Type: M I
```

Similarly, ***parity*** can be applied to lists and finite sets of integers. Since ***parity*** can only return one of two distinct values and finite sets do not contain duplicate entries, the resulting object can be smaller than the original argument.

```
parity ! [1..10]

   (4)  [1,0,1,0,1,0,1,0,1,0]
                                                    Type: L I


parity ! {1..10}

   (5)  {1,0}
                                                    Type: FSET I
```

The use of *!* is not restricted to unary functions.

```
gcd(2,!m)

       +1  2  1+
   (6) |       |
       +2  1  2+
                                                    Type: M I


gcd(2,! {1..10})

   (7)  {1,2}
                                                    Type: FSET I
```

Union and Any

The interpreter now supports the datatypes **Union** and **Any**. The domain constructor **Union** creates a *discriminated* union of several datatypes. The **case** infix operator is used to identify the branch in which the value of the union lies.

Type **Any** is rather special (at least among the datatypes usually employed by **Scratchpad II** users), in that its objects are self-type-identifying values: each object consists of a type and an object of that

type. Any object may be coerced to an object of type **Any**.

If the interpreter is given a list containing objects of several different domains, the resulting list will be of type **List Any**. In this way, one can construct "records" of arbitrary length, indexed by list position. (Of course, the normal **Record** is also available. Note that list indexing starts with *0*.)

```
h := [3,4/5,"foo",9.4]

                4
   (1)  [3,---,"foo",9.4]
                5

Type: L A

h.2

   (2)  "foo"

Type: A
```

The interpreter will automatically try to coerce an element of type **Any** to the associated domain of the object, if necessary.

```
h.3 ** 3

   (3)  830.584

Type: F
```

Robert S. Sutor
Martin L. Brock
Michael Lucks

Algebraic Facilities

This section discusses work recently completed or currently in progress to expand the library of algebraic facilities available in **Scratchpad II**.

The algebraic facilities are currently being reorganized. Although **Scratchpad II** provided many different multivariate polynomial constructors, operation names were not always consistent among them. In addition, some constructors had operations not present in others. We have standardized the names and implemented functions so that now, more than ever before, one can write algorithms involving multivariate polynomials that are not dependent on the particular choice of constructor. This allows one, for example, to test the appropriateness of a given polynomial representation for an algorithm. We are also standardizing the use of ***eval*** in the univariate and multivariate domains.

Moss Sweedler has been working to reorganize the basic categories underlying most of the algebraic constructors. This work takes advantage of several important features that will be present in the new **Scratchpad II** language (as embodied by the compiler being written by Stephen Watt). One such feature would allow different domains in a category to have different names for the same operation. A domain having the category **Group** would allow you to specify the name of the group operation, for example.

With the addition of Rüdiger Gebauer as a visitor to the group, work has begun on optimizing the Gröbner Basis packages. We are also working to connect these packages with those for solutions of equations so that one can solve solutions of nonlinear polynomial equations.

Michael Lucks is implementing several major improvements to the multivariate polynomial factorization packages. These modifications should significantly improve performance and permit the factorization of polynomials containing more variables than is currently feasible. If only a slight amount of sparsity exists among the factors, a generalized, more powerful technique for coefficient predetermination frequently obtains the complete factorization without performing additional Hensel lifting. An inexpensive method for detecting and correcting extraneous multivariate factors minimizes the extraneous factor problem without requiring multiple univariate factorizations.

A rudimentary domain of truncated power series has been implemented. It should soon be able to compute series expansions of functions defined by differential equations.

Vlad Grinberg has implemented factorization of Gaussian integers in the interpreter and will be adding that and related functions to the library of compiled **Scratchpad II** code.

Patrizia Gianni has completed a new vector space domain constructor that, for example, allows one to compute with univariate polynomials as elements of a vector space. Consider the vector space of all univariate polynomials in x with rational coefficients, and call it *vv*.

```
vv := VVectorSpace(RN,P[x] RN))

   (1)  VVS(RN,P[x]RN)
```

Declare *a, b, c* and *d* to be such polynomials and assign them values.

```
(a,b,c,d,e): P[x] RN

a:=2+x

   (3)  x + 2

b:=1-x**2

                2
   (4)  - x  + 1
```

```
c:=x**2+x-1

                 2
   (5)  x  + x - 1

d:=x**2-x

                 2
   (6)  x  - x
```

Let s be the subspace spanned by a and b, and t that spanned by c and d. One can compute $s \cap t$ and $s + t$.

```
s:=span([a,b])

                  2
   (8)  [- x  + 1,x + 2]

t:=span([c,d])

          2      2
   (9)  [x  - x,x  + x - 1]

intersection(s,t)

          2     1       - 3
   (10)  [x  + (---)x + -----]
                5         5

s + t

          2                   2
   (11)  [x  + x - 1,x + 2,- x  + 1]
```

Once so constructed, one can test element inclusion in a subspace. Here we ask if $3x^2 + x - 1$ is an element of s.

```
e:=3*x**2+x-1

                 2
   (12)  3x  + x - 1

element?(e,s)

   (13)  true

-- find the coordinate of e as element of s
coordinate(e,s)

   (14)  [- 3,1]
```

Though **Scratchpad II** users have been able to compute with elements of simple algebraic extensions for some months now, it has only been recently that code existed for the computation of partial derivatives. The following example computes y to be an element of the field extension of rational functions over the integers such that y satisfies the equation

$$y^3 + y + x = 0,$$

and then computes the partial derivative of y with respect to x.

```
y | y**3 + y + x = 0

   (1)  y

y**3 + y

   (2)  - x

pderiv(y,x)

   (3)
          - 6      2          9x          - 4
   (----------)y  + (----------)y +  ----------
              2                2              2
     27x  + 4         27x  + 4       27x  + 4
```

Other Work in Progress

- Extension of the Hensel GCD construction to work with more general coefficient rings.
- Integration of transcendental functions using the Risch algorithm and integration of genus 0 algebraic functions.
- Basic number theory and combinatoric packages including computation of such things of the Euler ϕ function, Möbius function, σ functions, τ function, binomial and multinomial coefficients, and partition functions.
- Completion of the ideal theory package.

Barry M. Trager
Robert S. Sutor

Joint Studies and the ALGEBRA 4381

Scratchpad II is being made available to the scientific community via a dedicated IBM 4381 at IBM Research in Hawthorne, New York. The machine can be accessed by direct dial-up or from CSNET/X25 and Arpanet. Via direct dial-up, users can run **Scratchpad II** from a variety of terminal types including IBM Personal Computers and other ASCII terminals (such as VT100s). Access is also provided to users who have direct access to Arpanet or CSNET/X25 host machines. Access directly from Telenet is expected in early 1986.

The following is a list of 50 initial **Scratchpad II** joint studies with university and research centers, many of which remain pending at the time of publication. The joint studies in Europe (those preceded by †) involve an early distribution of the system.

Mathematics Departments (28)

†Bath	James Davenport
Brandeis	Michael Stillman
Brown	Philip Davis
Columbia	David Bayer
Cornell	John Hubbard
Drexel	Jet Wimp
Harvard	John Tate
Hofstra	Harold M. Hastings
Hofstra	Jack Reichman
Illinois	John W. Gray
Kent State	Michael Rothstein

Kentucky	Paul Eakin
Michigan	Robert Griess
Minnesota	Mark Feshbach
North Carolina State	Michael Singer
†Paris	Daniel Lazard
Penn State	Edward Formanek
Penn State	George E. Andrews
†Pisa	Patrizia Gianni
Rockefeller	Joel Cohen
Rutgers	Eduardo D. Sontag
RIT	George Georgantas
RPI	M. S. Krishnamoorthy
UCSD	John J. Wavrik
Virginia	Leonard Scott
Wisconsin	Peter Orlik
Wisconsin	Richard Askey
York	Donald Solitar

Computer Science Departments (13)

Cornell	Dexter Kozen
CMU	A. Nico Habermann
Delaware	B. F. Caviness
Illinois	William Gear
MIT	Gail Zacharias
MIT	Richard Zippel
Rice	C. Sidney Burrus
Rice	Hans Boehm
SMU	David Y. Y. Yun
UC Berkeley	Richard Fateman
UCLA	Leonard Kleinrock
Waterloo	Bruce Char
Yale	Marina Chen

Government/Industrial Laboratories (7)

AT&T Bell Labs.	Andrew Odlyzko
Dept. of the Army	A. Brinton Cooper III
Dept. of the Navy Dahlgren, VA	Kenneth Bannister
†IMAG France	Jean Della Dora
Lawrence Livermore	Grant Cook
Los Alamos	William A. Beyer
Rand Corporation	A. C. Hearn

Engineering Departments (2)

Columbia	Morton B. Friedman
North Carolina State	Clarence Maday

If you or a colleague would like to avail yourself of **Scratchpad II**, please send me a letter describing the nature of the use you would make of the system. In turn, our group will try to arrange for a joint study to be set up between IBM and your department. We would only ask that you pay for communication costs as appropriate.

Because of the limited capacity of the 4381, we cannot promise that all requests to use our system will be accepted. We will try to accommodate as many as possible.

Richard D. Jenks

Algebra Snapshot: Linear Ordinary Differential Operators

Here we report on our work on operators in **Scratchpad II**. The package we describe provides linear ordinary differential operators. We decided to restrict our attention to this limited, well-understood domain for two reasons. First, this domain was a tool to be used in some work we were doing on ordinary differential equations. Second, we felt that if we could create the differential operator domain using only user-level facilities, then it could be used as an example for other operator domains.

As far as we are aware, operator algebra has not been part of the initial design of any computer algebra system. After-the-fact addition of classes of operators can be awkward, requiring either special functions for operator manipulation ("opPlus", "opTimes", etc.), which is ugly, or requiring modification of the base system, which is impossible for most users.

Scratchpad II addresses this issue by providing generic operations and by making application itself an operation. For example, the domain **Mapping**(A, B) of functions from A to B exports the application operation

$$".": (\textbf{Mapping}(A,B), A) \to B$$

which takes a mapping from A to B and an element of A to produce an element of B.

The notion of generic application is fundamental to the system and is used not only for function invocation but also for subscripting arrays, list element extraction, record field selection and indexing tables. Two syntaxes are provided for application, one associating to the left and another to the right:

f a b means f applied to (a applied to b)
f.a.b means (f applied to a) applied to b

To create a ring R of operators on a set S in **Scratchpad II**, one adds the operation

$$".": (R, S) \to S$$

to the constructor of R.

LODO

LODO(A,M) is the domain of linear ordinary differential operators over an A-module M, where A is a differential ring. This includes the cases of operators which are polynomials in D acting on scalars or vectors depending on a single variable. The coefficients of the operator polynomials can be integers, rational functions, matrices or elements of other domains.

Differential operators with constant coefficients

We begin by making some type assignments and declaring *Dx* to be a linear differential operator. Please refer to "Some Scratchpad II Constructor Name Abbreviations" on page 8 for expansions of the type abbreviations.

```
PZ := UPDR(P[x]I, I);

PQ := UPDR(P[x]RN,RN);

Dx: LODO(RN, PQ)
```

PZ is the domain of univariate polynomials in *x* over the integers with some additional properties that make it into a differential ring. Similarly, **PQ** is a domain of univariate polynomials over the rationals. (Within a few months one should be able to use **P[x] I** and **P[x] RN** directly, without resorting to the **UPDR** constructor.)

Operators are created as polynomials in *D()*.

```
Dx := D()

   (5)  "D"

a  := Dx  + 1

   (6)  D + 1

b  := a + 1/2*Dx**2 - 1/2

                1  2      1
   (7)  (---)D  + D + ---
          2           2

c := (1/9)*b*(a + b)**2

   (8)
        1    6      5    5      13   4      19   3
      (----)D  + (----)D  + (----)D  + (----)D
        72          36          24          18
  +
        79   2      7         1
      (----)D  + (----)D + ---
        72          12        8
```

To apply the operator *a* to the value *p* the usual function call syntax is used.

```
p: PQ := 4*x**2 + 2/3;

a p

                  2          2
   (10)  4x  + 8x + ---
                     3
```

Operator multiplication is defined by the identity $(a \times b)p = a(b(p))$.

```
(a*b) p = a b p

                 2          37       2          37
   (11)  2x  + 12x + ---- = 2x  + 12x + ----
                     3                   3
```

Operator expressions may be applied directly.

```
(a**2 - 3/4*b + c) (p + 1)

                2      44         541
   (12)  3x  + (----)x + -----
                  3        36
```

When the operator coefficients are rational, it is possible to factor.

```
factor c

                1          2        4
   (13)  (----)(D + 3) (D + 1)
               72
```

Differential operators with rational function coefficients

We clear the values of most of the variables, but retain those of **PZ** and **PQ**.

```
)clear value Dx a b p e f
(Dx, a, b): LODO(QF PZ, QF PZ)


Dx := D()

   (16)  "D"

b := 3*x**2*Dx**2 + 2*Dx + 1/x

              2 2          1
   (17)  3x D  + 2D + ---
                       x


a := b*(5*x*Dx + 7)

              3 3          2          2          7
   (18)  15x D  + (51x  + 10x)D  + 29D + ---
                                         x


p: QF PZ := x**2 + 1/x**2

             4
            x  + 1
   (19)  --------
                2
               x
```

Since the operator coefficients depend on *x* the operator mulitplication is not commutative.

```
(a*b - b*a) p
```

$$(20)\quad \frac{-75x^4 + 540x - 75}{x^4}$$

When the coefficients come from a field it is possible to define left and right division of the operators. The results of ***ldiv*** and ***rdiv*** are quotient/remainder pairs.

```
ldiv(a,b)
```

$$(21)\quad [5x*D + 7, 0]$$

```
-- "%" means the result of the previous expression
a - (b * %.quotient + %.remainder)
```

$$(22)\quad 0$$

```
rdiv(a,b)
```

$$(23)\quad [5x*D + 7, 10D + \frac{5}{x}]$$

```
a - (%.quotient * b + %.remainder)
```

$$(24)\quad 0$$

The divisions allow the computation of left and right greatest common divisors via remainder sequences, and consequently the computation of left and right least common multiples.

```
e := lgcd(a,b)
```

$$(25)\quad 3x^2D^2 + 2D + \frac{1}{x}$$

A GCD doesn't necessarily divide a and b on both sides—here the left GCD does not divide *a* on the right.

```
ldrem(a, e)   -- the remainder from left division
```

$$(26)\quad 0$$

```
rdrem(a, e)   -- the remainder from right division
```

$$(27)\quad 10D + \frac{5}{x}$$

Likewise, an LCM is not necessarily divisible from both sides.

```
f := rlcm(a,b)
```

$$(28)\quad 20x^5D^5 + (\frac{684x^4 + 80x^3}{3})D^4 + (\frac{5832x^3 + 1656x^2 + 80x}{9})D^3 + (\frac{3672x^2 + 2040x + 352}{9})D^2 + (\frac{172}{9x})D + \frac{-28}{9x^2}$$

```
rdrem(f, b)
```

$$(29)\quad 0$$

```
ldrem(f, b)
```

$$(30)\quad (\frac{-1176x + 160}{9x})D + \frac{312x - 80}{9x^2}$$

Differential operators with elementary function coefficients

Problem: find the first few coefficients of $e^x x^{-i}$ in $L_3\,\phi$, where

$$L_3 = D^3 + \frac{G}{x^2}D + \frac{H}{x^3} - 1$$

and

$$\phi = \sum_{i=0}^{\infty} \frac{s_i e^x}{x^i}$$

We will compute the first five coefficients. We start by defining the domain **EDF** to be the elementary functions viewed as a differential ring in the variable *x*.

```
)clear value n Dx L3 phi rho num xnum
n := 5
```

$$(32)\quad 5$$

```
EDF := ODR(SE, EF P I, x);
```

Next we assign the differential operator *L* and the first terms of the function ϕ.

```
(Dx, L3): LODO(EDF, EDF)
Dx := D()
```

$$(35)\quad \text{"D"}$$

```
L3 := Dx**3 + G/x**2*Dx + H/x**3 - 1
```

$$(36)\quad D^3 + (\frac{G}{x^2})D + \frac{H - x^3}{x^3}$$

```
phi := +/[ s[i]*exp(x)/x**i for i in 0..n ]
```

$$(37)\quad \frac{(s_5 + x s_4 + x^2 s_3 + x^3 s_2 + x^4 s_1 + x^5 s_0)\exp(x)}{x^5}$$

```
rho: EF P I := L3 phi
```

(38)

$$\begin{aligned}
&\Big((H + (x-5)G - 15x^2 + 90x - 210)s_5 \\
&+ (x H + (x^2 - 4x)G - 12x^3 + 60x^2 - 120x)s_4 \\
&+ (x^2 H + (x^3 - 3x^2)G - 9x^4 + 36x^3 - 60x^2)s_3 \\
&+ (x^3 H + (x^4 - 2x^3)G - 6x^5 + 18x^4 - 24x^3)s_2 \\
&+ (x^4 H + (x^5 - x^4)G - 3x^6 + 6x^5 - 6x^4)s_1 \\
&+ (x^5 H + x^6 G)s_0 \Big) \\
&* \exp(x) \\
&/ x^8
\end{aligned}$$

We extract the numerator and remove the common factor of *exp(x)*.

```
-- ending the next line with ";" suppresses output
num: P I := rho / exp(x) * x**(n+3);
```

To collect terms with like powers, make x the main variable.

```
xnum: P[x] P I := num;
```

The answer is obtained by reading off the coefficients corresponding to the powers $\frac{1}{x^0}$ to $\frac{1}{x^n}$.

```
[ coef(xnum, n+3 - i) for i in 0..n ]
```

(41)

$$[0,\ 0,\ -3s_1 + s_0 G,\ -6s_2 + s_1 G + 6s_1 + s_0 H,$$
$$-9s_3 + s_2 G + 18s_2 + s_1 H - s_0 G - 6s_1,$$
$$-12s_4 + s_3 G + 36s_3 + s_2 H - 2s_2 G - 24s_2]$$

Differential operators with matrix coefficients acting on vectors

Define the differential ring of 3 by 3 matrices of polynomials in x over the integers and the three dimensional vector space of these polynomials.

```
)clear value Dx a b t p0 p1 p2 p q m

Mat  := SMDR(3, PZ);

Vect := DPMM(3, PZ, Mat, PZ);
```

The matrix *m* will be used as a coefficient and the vectors *p* and *q* will be operated upon.

```
t: PZ := x**2;

m := zero(3,3)$Mat;


m(0,0) := t;
m(0,1) := 1;
m(1,0) := 1;
m(1,1) := t**2;
m(2,2) := 4*t;


m
```

$$(52)\quad \begin{bmatrix} x^2 & 1 & 0 \\ 0 & 4 & 0 \\ 1 & x & 0 \\ 0 & 0 & 4x^2 \end{bmatrix}$$

```
p0: PZ := 3*x**2 + 1;
p1: PZ := 2*x;
p2: PZ := 7*x**3 + 2*x;

p: Vect := [p0, p1, p2]$V(PZ)
```

$$(56)\quad [3x^2 + 1, 2x, 7x^3 + 2x]$$

```
q: Vect := m * p
```

$$(57)\quad [3x^4 + x^2 + 2x, 2x^5 + 3x^2 + 1, 28x^5 + 8x^3]$$

The operation

"*": (Mat, Vect) → Vect

is defined so it makes sense to define operators with coefficients in ***Mat*** acting on members of ***Vect***

```
(Dx, a, b): LODO(Mat, Vect)

Dx := D()
```

(59) "D"

```
a := Dx  + m
```

$$(60)\quad D + \begin{bmatrix} x^2 & 1 & 0 \\ 0 & 4 & 0 \\ 1 & x & 0 \\ 0 & 0 & 4x^2 \end{bmatrix}$$

```
b := m*Dx  + 1
```

$$(61)\quad \begin{bmatrix} x^2 & 1 & 0 \\ 0 & 4 & 0 \\ 1 & x & 0 \\ 0 & 0 & 4x^2 \end{bmatrix} D + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
a * b

  (62)
      ┌ 2          ┐
      │x    1    0 │
      │            │2
      │     4      │D
      │1    x    0 │
      │            │
      │           2│
      └0    0   4x ┘
  +
      ┌ 4                  4    2                            ┐
      │x   + 2x + 2       x  + x                   0          │
      │                                                       │
      │  4     2        8      3                              │D
      │ x  + x         x  + 4x  + 2                0          │
      │                                                       │
      │                                        4              │
      └     0                  0            16x  + 8x + 1     ┘
  +

      ┌ 2          ┐
      │x    1    0 │
      │            │
      │     4      │
      │1    x    0 │
      │            │
      │           2│
      └0    0   4x ┘

(a+b) (p + q)

  (63)
        6      5      4      3      2
   [3x  + 14x  + 17x  + 22x  + 10x  + 18x + 6,

      9      8      6      5      4      3     2
    2x  + 10x  + 3x  + 10x  + 16x  + 12x  + 7x  +

                      7       6      5       4
    18x + 6,     112x  + 560x  + 88x  + 320x  +

       3      2
    23x  + 53x  + 2x + 2]
```

Stephen M. Watt
Jean Della Dora

Current Visitors to the Scratchpad II Group

Martin Brock
A recent M.S. graduate in Electrical Engineering from M.I.T.. System: 5/85-7/86

Rüdiger Gebauer
Wiss. Ang., Inst. f. Angewandte Mathematik, Universität Heidelberg, Im Neuenheimer Feld 294, D 6900 Heidelberg. Interface: 12/85-11/86.

Professor Patrizia Gianni
Dipartimento di Matematica, Universita Di Pisa, Pisa, Italy. Algebra: 1/86.

Vladimir A. Grinberg
Graduate Student, Columbia University, New York, New York. Interpreter: 12/85-6/86.

Michael Lucks
An M.S. graduate in Computer Science from the University of Hawaii. Interpreter: 2/85-9/86.

Mohamed Mobarak
Undergraduate student, Department of Computer Science, Cornell University, Ithaca, New York. System: 1/86.

Professor Moss E. Sweedler
Department of Mathematics, Cornell University, Ithaca, New York, Interface: 9/85-6/86.

Recent Demonstrations of Scratchpad II

August, 1985	IJCAI '85, Los Angeles, California.
August, 1985	AMS Summer Conference in Computational Number Theory, Humboldt State University, Arcata, California.
September, 1985	CIRM Conference on Computers in Algebraic Geometry, Trento, Italy.
October, 1985	NSF Workshop on Comp. in Algebra and Number Theory, University of California, Berkeley.
November, 1985	Computer Science Department Chairmen's Conference, Yorktown Heights, New York.
December, 1985	Future Directions in Computing, Grenoble, France.

For your information ...

The **Scratchpad II** Computer Algebra System currently runs only under the IBM **VM/SP** operating system on mainframe computers. A three megabyte segment is shared among all users on a given computer, and each user requires at least a four megabyte virtual machine.

At last count, the **Scratchpad II** library consisted of 53 category constructors, 102 domain constructors and 59 package constructors.