THE UCLA COMPUTER SCIENCE DEPARTMENT QUARTERLY

THE DEPARTMENT AND ITS PEOPLE

FALL 1987/WINTER 1988

VOL. 16 NO. 1

SCHOOL OF ENGINEERING AND APPLIED SCIENCE

Subscriptions to this publication are available at $20.00/year (three issues) by contacting Ms. Brenda Ramsey, UCLA Computer Science Department, 3713 Boelter Hall, Los Angeles, CA 90024-1596.

THE UCLA COMPUTER SCIENCE DEPARTMENT QUARTERLY

The Computer Science Department And Its People

Fall Quarter 1987/Winter Quarter 1988
Vol. 16, No. 1

SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF CALIFORNIA, LOS ANGELES

Fall-Winter CSD Quarterly

CONTENTS

TECHNICAL PAPERS

COMPUTER SCIENCE AT UCLA AND COMPUTER SCIENCE IN THE UNITED STATES

Gerald Estrin, Chair

The Fall-Winter issue of the Computer Science Department Quarterly discusses the state of the Department and its people in the context of the university, profession and the outside world.

Recently, the Computing Research Board sent me the 1986-87 Taulbee Survey Report on the Production and Employment of Ph.D.'s and Faculty in Computer Science and Engineering. Every year there is a survey of Ph.D. granting Computer Science Departments throughout the United States. It has been called the Taulbee Survey Report in honor of Orrin E. Taulbee of the University of Pittsburgh who dedicated himself to these surveys from 1970 to 1984. This year, through strenuous efforts by David Gries and Dorothy Marsh of the Cornell University Computer Science Department, the 1986-87 survey has become available in record time. That survey provides important context for our activities.

Immediately below, we provide some metrics for the research environment in the UCLA Computer Science Department which were gathered in September 1987 for the National Science Foundation. The 1986-1987 Taulbee Survey Report by the Computing Research Board, reprinted with permission of the authors, follows the metrics.

DEPARTMENTAL DATA		UNDERGRADUATE DATA	
	September 87		September 87
Number of FTE Faculty	35.25	Number in Lower Division	98
Regular Faculty	31.25	Number in Upper Division	174
Adjunct Faculty	.63	Planning Graduate Study	31%
Research & Post-doc Faculty	0	Planning Employment	69%
Instructors	3.37	Average Starting Salary	$28,944.00
		Average Number of Offers	3
Number of Technical Support Staff	10.0		
Programming & Systems	7.0	GRADUATE DATA	
Hardware & Electronics	3.0		
Other	0	Number of Part-Time M.S. Students	0
		Number of Full-Time M.S. Students	114
Number Administrative Support Staff	25.0	Number of Part-Time Ph.D. Students	0
Percentage Budget		Number of Full-Time Ph.D. Students	159
Instruction	31%	Average Entering GRE Scores	
Internally Funded Research	3%	Verbal	69.52
Externally Funded Research	66%	Quantitative	95.34
Other	0%	Advanced	84.75
		Average Entering Undergraduate GPA	3.62
Total Department Budget	$12,022,000.00	Out of	4.0
Capitalization	$ 8,049,938.00	Ratios	
Total Space/Faculty Member		Number Accepted/Number Applying	175/506
Office	242 sq.ft	Number Enrolled/Number Accepted	80/175
Laboratory	293 sq.ft	Numbers Holding:	
Research		Fellowships	14
Number External Grants & Contracts	110	RA's in Submitting Department	93
Current Year Total Research Budget		RA's in Other Departments	0
Internal Funds	$ 350,028.00	TA's in Submitting Departments	18
External Funds	$ 4,119,774.00	TA's in Other Departments	5

The 1986-87 Taulbee Survey Report[0]
The Computing Research Board's Survey

on the

Production and Employment of Ph.D.'s and Faculty in

Computer Science and Engineering[1]:

David Gries, Dorothy Marsh

Computer Science, Cornell University

January 1988

Introduction

This report describes the results of a survey completed in December 1987 on the production and employment of Ph.D.'s and faculty of Ph.D.-granting Computer Science/Engineering Departments during the academic year 1986-87.[2] All 123 Computer Science (CS) departments (111 U.S. and 12 Canadian) participated. In addition, for the first time, 22 departments offering the Ph.D. in Computer Engineering (CE) were included.[3] Throughout this report, CE statistics are reported separately so that comparisons with previous years can be made for CS, but the intention is to merge all statistics for CS and CE after several years. Some highlights from the survey are:

- The 123 CS departments produced 466 Ph.D.'s, an increase of 13% over the previous year; 255 were Americans, 19 Canadians, and 181 foreign (11 were unknown). Of the 466, 232 went to academia, 151 to industry, 8 to government, and 34 overseas; 4 were self-employed (37 were unknown).
- The 123 CS departments expect to produce 707 Ph.D.'s next year. (This is far too optimistic; we expect, instead, another increase of 13% to 525.)
- 1008 students passed their Ph.D. qualifying exam in CS departments, an increase of 17% over 1985-86.
- The 123 CS departments have 2269 faculty members, an increase of almost 11%: 857 Assistant, 630 Associate, and 782 Full Professors.
- The 123 CS departments reported hiring 238 faculty and losing 179 (to retirement, death, other universities, graduate school, and non-academic positions).
- The 123 CS departments want to grow from 2325 faculty members (including visitors and non-tenure-track faculty) to 3133 in five years, an increase of 35%, at an average rate of 1.3 per department per year. (Last year, they wanted a growth of 1.4 per department but grew 1 per department.)

---

[0]The title of the survey honors Orrin E. Taulbee of the University of Pittsburgh, who conducted these surveys for the Computer Science Board annually from 1970 to 1984.

[1]The 145 departments completing the questionnaire deserve a round of applause.

[2]123 departments reported on an academic-year basis and 13 on a 1987 calendar-year basis; 9 neglected to indicate the system used. Departments usually report on the same basis from year to year, so the difference has no effect when viewing changes in survey data.

[3]The Forsythe list — the list of all departments in the U.S. and Canada that grant a Ph.D. in CS or CE — is maintained by Terry Walker, a member of the Computing Research Board. This is first year that the CE departments have been included.

One can draw some conclusions and make some predictions. First, the growth of 13% in CS Ph.D. production, although less than the 20% increase of 1985-86, is encouraging. Because of the increase in Ph.D. qualifying exam passage, this growth is expected to continue, and we can expect 600 Ph.D.'s per year by 1991.

The field continues to be far too young and inexperienced, a problem that only time can solve. CS continues to have more assistant professors than full professors, which puts an added burden on the older people. In fact, the ratios of assistant and associate professors to full professors has not changed appreciably in three years. No other field, as far as we know, has this problem — in fact, most scientific fields are 80 to 90 percent tenured in many universities. The CE departments have more full professors than assistant professors, mainly because many are older EE departments offering CE degrees.

The increase in Ph.D. production over last year is reportedly made up entirely of U.S. citizens, and the percentage of CS Ph.D.'s given to foreign students fell from 46 to 40. Whether this is a trend remains to be seen.

Last year, 118 Ph.D.-granting CS departments were identified; this year, 123. The growth in the number of such departments is slowing down and, we believe, will soon essentially halt. The Computing Research Board was able to identify 33 departments giving a Ph.D. in CE (but not in CS), and they were asked to participate in this survey. Whether there will be more growth in CE departments is not known.

Some methodological comments

Questionnaires were sent to 123 CS Ph.D.-granting departments and 33 CE Ph.D.-granting departments in late October 1987. (The titles of the departments appear in Table 0).

Table 0. Titles of Departments

Number of departments	Title
91	Computer Science(s)
17	Electrical and Computer Engineering
11	Computer and Information Science(s)
6	Electrical Engineering and Computer Science
6	Computer Science and Engineering
2	Computer Engineering
2	Electrical Engineering
2	Information and Computer Science
1	Advanced Computer Studies
1	Applied Sciences
1	Computational Science
1	Computer Science and Electrical Engineering
1	Computer Engineering and Information Science
1	Computer Engineering and Science
1	Computing Science
1	Mathematical Sciences
(Instead of 'Department', the terms 'Center', 'Division', 'Program', and 'School' were each used at least once.)	

With the help of 268 phone calls, all 123 CS departments and 22 of the 33 CE departments had completed the questionnaire by 10 December 1987. Thus, the figures in this report are complete for CS. We hope to have a better response from the CE departments in next year's survey. The accuracy of this report depends, of course, on the accuracy with which the questionnaires were

filled out by the individual departments. The new electrical engineering departments giving a Ph.D. in CE had a more difficult time completing the questionnaire, for they were asked to give information only on the CE part of their departments, and the required information was difficult to extract.

As with most surveys, a small part of the data in the survey was not filled in or obviously was incorrectly entered. We took the liberty to adjust some figures and estimate a few others — for example, in a few cases, with 142 or 143 out of 145 departments reporting a figure in a field, we estimated that field for the others. Our goal was to make this report consistent, clear, and simple, without modifying the overall results in any way.

In some places, we analyze the data for the higher ranked departments as compared to the lower ranked and unranked ones, using for ranking the 1980 survey done under the auspices of the National Research Council [0]. (We also included the two largest Canadian universities somewhere within the top 20.) Survey [0] is now eight years old, and many changes have occurred in CS since then (e.g. the emergence of over 60 Ph.D.-granting CS departments); nevertheless, this breakdown still provides some useful comparisons.

From time to time within this report, in order to draw meaningful conclusions regarding growth of the field (using older surveys), we compare figures for the CS departments only, keeping figures for CE separate. Throughout this report, figures for 1984-85 are taken from [1], for 1985-86 from [2], and for 1970-84 from [3]. The figures for 1970-84 may not be accurate because not all departments completed questionnaires in those days. Finally, for comparisons with graduate study in mathematics, figures are taken from [4].

Data on Students

Ph.D. production and its growth

The field of CS produced 466 Ph.D.'s in 1986-87, an increase of 54 (13%) over 1985-86 and an increase of 236 (103%) over 1980. The figures on Ph.D. production for CS and CE, as well as for qualifying-exam passage and sizes of incoming classes, are given in Table 1. In Table 1, in the column headed 'No. of depts', the first number is the number of departments reporting and the second the total number of known Ph.D.-granting departments.

Table 1. Ph.D. Production and its Growth

	Year	No. of depts	Ph.D.'s produced	average per dept	Qualifying exam passage	average per dept	New Ph.D. students	average per dept
CS	1980-81		230					
CS	1984-85	103 (109)	326	3.2	755	8.21	1177	12
CS	1985-86	117 (118)	412	3.5	858	7.30	1170	10
CS	1986-87	123 (123)	466	3.8	1008	8.19	1430	12
CS-CE	1986-87	145 (156)	559	3.9	1168	8.05	1621	11
Math	1986-87	259 (317)	845	3.3				

CS produces more Ph.D.'s per department than math does at this point, although CS has 18.5 faculty per department and math has 30.2. Further, let us consider only associate and full professors, as the producers of most of the Ph.D.'s. In CS, the average CS associate and full professor produced 0.33 Ph.D.'s in 1986-87; the average math associate and full professor, 0.18.

As mentioned earlier, CS Ph.D.-production increased 13% this year and 20% last year. Future growth is expected. Indeed, the 123 departments project 707 Ph.D.'s in 1987-88 — a 52% increase! A more realistic estimate is another 13%, to 525. As some evidence for our estimate, in the last survey the departments optimistically predicted 652 Ph.D.'s, we predicted 480, and 466 were produced.

Future increases in Ph.D. production are a matter of concern to the field. Estimates of the annual need for new Ph.D.'s range from 600 to over 1,000, and the field is growing steadily to meet the need. However, growth in Ph.D. production requires a commensurate growth in funding for research. Because of this interest in Ph.D. production, we go into more detail.

In 1986-87, an average of 3.9 CS-CE Ph.D.'s were produced per department, (see Table 2) with 27 departments producing 0, 20 producing 1, 23 producing 2, and 25 producing 3. Thus, 95 departments produced less and 50 departments more than the average. The 50 that produced more than the average — roughly one third of the departments — produced 75% of the Ph.D.'s.

The over-average group of 50 expects to increase its Ph.D. production in one year far less (by 115, or 28%) than the under-average group (by 164, or 116%). The expected growth has remained about the same for the over-average group (24% in both 1985-86 and 1984-85). But the predicted one-year growth by the under-average group was 167% in 1984-85, 164% in 1985-86, and 116% in 1986-87.

In an effort to find different expected-growth patterns, the data for the groups of departments in various rankings (according to [0]) is presented in Table 2.

Twenty-two CS-CE departments had 15 or more students passing the qualifying examination; they accounted for 55% of the students passing the exam.

Table 2. Ph.D. Production by Ranking in 1986-87

Rank dept	Ph.D.'s produced	average per dept	Ph.D.'s next yr	average per dept	Qualifying exam passage	average per dept	New Ph.D. students	average per
CS (all)	466	3.8	707	5.8	1008	8.3	1430	11.8
CS 1-12	166	13.8	196	16.3	255	21.2	287	23.9
CS 13-24	66	5.5	123	10.3	142	11.8	207	17.3
CS 25-36	66	5.5	103	8.6	143	11.9	176	14.7
Other CS	168	1.9	285	3.3	468	5.5	760	8.9
CE	93	4.2	131	6.0	160	7.3	191	8.7

Sex and minority status of the Ph.D.'s.

Table 3 gives the figures on Ph.D.'s awarded to minority students and females. The figures are rather depressing from the standpoint of minority and female representation in the field. Table 4 shows the statistics since 1970, with the data before 1984-85 being taken from [3]. Throughout the 1980's the percentage of Ph.D.'s who are women has stayed relatively constant at about 11%, blacks at 1%, and Hispanics at 2%.

Table 3. Sex and Minority Status of the Ph.D.'s

	CS			CE			CS-CE		
Ph.D. Minority Status	Male	Female	Total	Male	Female	Total	Male	Female	Total
White, not of hispanic origin	263	39	302	46	2	48	309	41	350
Black, not of hispanic origin	1	0	1	2	0	2	3	0	3
Hispanic	8	0	8	1	0	1	9	0	9
Other	143	12	154	41	1	42	184	13	196
Total	415	51	466	90	3	93	505	54	559

Table 4. Sex, Minority Status, and Citizenship of the Ph.D.'s since 1970						
Year	70 71 72 73 74	75 76 77 78 79	80			
81 82 83 84	84-85	85-86	86-87			
Total	112 124 206 208 203	256 246 208 223 248	230 235 244			
256 274	326	412	466			
Female	1 4 12 7 6	21 14 14 19 24				
28 26 27 31 29	32	50	51			
Percent	1 3 6 3 3	8 6 7 9				
10	12 11 11 12 10	10	12	11		
Black	1 1 2 2 2	1 0 0 2				
1	0 0 1 2 3	3	6	1		
Percent	1 1 1 1 1	0 0 0 1				
0	0 0 0 1 1	1	1	0		
Hispanic	No information	No information	No information	7	6	8
Percent				2	1	2
Foreign	22 21 39 41 46	68 57 68 51 65				
82 79 83 86 87	122	184	181			
Percent	20 17 19 20 23	27 23 33 23 26				
36 33 34 34 32	37	45	40			

Citizenship of the Ph.D.'s

The number of Ph.D.'s given to foreigners remained essentially the same (in CS, 184 last year and 181 this year). Hence, the increase in Ph.D. production was all in American and Canadian citizens, and the percentage of Ph.D.'s to foreigners dropped from 46% to 40%. Figures for citizenship of the Ph.D.'s are given in Table 5. Table 4 contains the figures for foreigners from 1970 to 1987.

Table 5. Citizenship of the Ph.D.'s

	U.S.	Canadian	Foreign	Percent foreign	Unknown
CS	255	19	181	40%	11
CE	41	0	46	53%	6
CS-CE	296	19	227	42%	17

Employment of the Ph.D.'s

As shown in Table 6, in CS, 35% of the Ph.D.'s produced took positions in the U.S. or Canada outside academia, 54% took faculty positions in the U.S. or Canada, 8% took positions in other countries, and 8% were unknown. There is little change from last year, when the figures were 42%, 48%, and 9%.

Table 6. Employment of the Ph.D.'s

	Number of Ph.D.'s	Self-employed	Academia Ph.D. dept	Academia not Ph.D. dept	Academia not CS or CE	Industry	Government	Outside U.S. and Canada	Unknown
CS	466	4	177	41	14	151	8	34	37
CS-CE	559	4	194	44	19	183	18	39	58

Undergraduate and Masters degrees

Many universities and colleges have undergraduate and/or masters programs but do not award the Ph.D., so the data given below says little about the field of computer science as a whole.

Table 7 gives statistics on undergraduate and Master's degrees in Ph.D. departments, with columns labeled '87-88' representing expectations. CS undergraduate degrees dropped 4% this year, and the departments expect another 3% decrease next year.

Table 7. Undergraduate and Master's Degrees

Non-Ph.D. Degrees, Ph.D. departments only		Undergraduate				Master's			
		84-85	85-86	86-87	87-88	84-85	85-86	86-87	87-88
CS	Number of degrees	10422	10947	10540	10247	2889	3720	3614	3751
3751									
	Number of depts. responding	96	116	121	120	101	116	12	
3	122								
	Average per dept.	109	94	87	85	29	32	29	
31									
CE	Number of degrees			2103	2147			731	787
	Number of depts. responding			22	22			22	22
	Average per dept.			96	98			33	36
CS-CE	Number of degrees			12643	12394			4345	4538
	Number of depts. responding			143	142			145	144
	Average per dept.			88	87			30	32

New graduate students in Fall 1987

Table 8 gives enrollment figures for new students in Fall 1987. In the table, 'Ph.D. program' stands for the number of new graduate students in Ph.D. programs, regardless of whether they intend to earn a Masters degree first. The number of new graduate students in CS is down slightly from last year (from 3722 to 3644), but the number of new graduate students in a CS Ph.D.-program rose from 1170 to 1430 (22%), another reason for expecting future growth in Ph.D. production.

The data for part-time Masters students needs some explanation. 50 departments had zero part-timers and 97 departments had 5 or fewer. For these departments, the part-time masters program may be inconsequential — perhaps just a small employee degree program of the University. On the other hand, the two largest part-time Masters programs had 205 and 152 new part-timers, respectively. The last column gives figures only for departments with between 6 and 50 new part-time masters students.

Table 9 gives the number of new Ph.D. students in CS departments this year and last, with departments grouped by rank.

Table 8. New Graduate Students in Fall 1987							
New Graduate Students		Total new graduate students	With CS degrees	Ph.D. program	Master's only program	Part-time Masters students	Part-time Master's in departments with 6-50
CS	Total	3644	1621	1430	2083	1078	495
	Depts. responding	121	117	121	121	120	33
	Average per dept.	30	14	12	17	9	15
CE	Total	952	128	191	556	481	75
	Depts. responding	22	22	22	22	22	5
	Average per dept.	43	6	9	25	22	15
CS-CE	Total	4596	1749	1621	2639	1559	570
	Depts. responding	143	139	143	143	142	38
	Average per dept.	32	13	11	18	11	15

Table 9. New Ph.D. Students in CS Departments							
Departments	Number of departments	Total			Average		
		1985	1986	1987	1985	1986	1987
Ranked 1-12	12	349	290	287	29	24	24
Ranked 13-24	12	219	176	207	18	15	17
Ranked 25-36	12	144	165	176	12	14	15
All other	62,81,85	465	678	760	8	7	9

Faculty

Table 10 contains statistics on departmental faculty in September 1987. In the Table, all figures are in terms of 'Full-time equivalents'. For example, two half-time appointments count as one position.

CS saw little change over last year in the proportions of faculty at the three levels. CS remains a relatively young field, with fewer full professors (6.4) than assistant professors (7) per department. The top 25 departments have about the same number (10 and 9.9) of full professors and assistant professors per department.

Table 10. Faculty Statistics, 1987-88 academic year								
Faculty Depts.	All CS-CE Depts.		123 CS Depts.		Top 25 CS Depts.		Other 98 CS	
	Total	Average	Total	Average	Total	Average	Total	Average
Tenure-track faculty	2702	18.6	2269	18.4	649	26.0	1620	16.5
Assistant professor	968	6.7	857	7.0	247	9.9	610	6.2
Associate professor	765	5.3	630	5.1	153	6.1	477	4.9
Full professor	969	6.7	782	6.4	249	10.0	533	5.4
Non-teaching research faculty	140	1.0	126	1.0	69	2.8	57	0.6
Postdocs	96	0.7	81	0.7	51	2.0	30	0.3
Non-tenure-track teachers	473	3.3	390	3.2	95	3.8	295	3.0
Other faculty (e.g. visitors)	281	1.9	201	1.6	60	2.4	141	1.4

Hiring for 1987-88

The CS-CE departments reported hiring 275 new faculty — 1.9 per department. CS departments in the U.S. hired 211 — again, 1.9 per department. Salaries were reported for new Ph.D.'s hired for Fall 1987 by 83 U.S. CS-CE departments, 69 U.S. CS departments, and 7 Canadian departments. Table 11 gives this salary information. The data for the Canadian universities are

shown separately in the table, in Canadian dollars. Canadian salaries are on a 12-month scale, the Canadian and U.S. dollars are different, and there are differences in the amount of consulting that typically can be performed.

The average U.S. salary for a new Ph.D. has increased from $36,668 in Fall 1985 to $38,957 in Fall 1986 (6.2%) to $40,885 in Fall 1987 (4.9%).

More information is included in Table 12, which gives the number of departments averaging a salary in each $1,000 range for Fall 1987 and two previous years (numbers are rounded and presented in thousands of dollars).

The departments reported hiring 29 faculty with Ph.D.'s earned in 1982 or later in a field other than computing science/engineering. The fields were: mathematics (3), electrical engineering (21), sociology, philosophy, physics (2), and management sciences. Part of the increase of new faculty with electrical engineering degrees is due to the inclusion of the CE departments in this year's survey.

Table 11. New Ph.D. Salaries, for Fall 1987

	All U.S. CS-CE depts.	All U.S. CS depts.	Top 24 U.S. CS depts.	Other 99 U.S. CS depts.	12 Canadian CS depts.
Total hired	248	211	65	146	27
Number of departments reporting salaries	83	69	18	51	7
Minimum	$35,000	$35,000	$37,333	$35,000	$38,500
Average (of the averages)	$40,885	$40,840	$41,540	$40,592	$42,440
Maximum	$52,100	$48,000	$47,000	$48,000	$48,578

Table 12. New Ph.D. Salaries, for Fall 1987 and Two Previous Years

Salary (in thousands)	35	36	37	38	39	40	41	42	43	44	45	46	47	48
1985-86: Number of depts.	2	10	11	11	5	5	1	0	0	0	0	0	0	0
1986-87: Number of depts.	3	1	9	11	16	14	5	4	2	0	0	0	0	0
1987-88: Number of depts.	1	1	1	3	8	13	14	20	4	1	1	0	1	1

Faculty salaries

Table 13 summarizes 9-month faculty salaries in U.S. departments during the 1987-88 academic year. The second column of each table gives the number of faculty (in each rank) for which salaries were reported and, in parentheses, the total number of faculty in that rank.

Departments reported the minimum, mean, and maximum salaries of assistant, associate, and full professors and the number of faculty in each rank. For minimum salaries (and for maximum salaries), the table shows the minimum, average, and maximum. Finally, the average is given over all salaries in each faculty rank — this is not the average of the means, but the true average.

Comparing this year's figures with last year's, we find that the average Assistant Professor salary rose 6.1% from $39544 to $41945. the average Associate Professor salary rose 5.3% from $45062 to $47428, and the average Full Professor salary rose 5.9% from $59503 to $63037.

Tables 14-17 supply the same information as Table 13, but for departments grouped by rank. Table 18 gives salary information for the CE departments. Table 19 gives salary information for the 11 Canadian departments that gave salary information. Table 20 gives the information for all U.S. CS and CE departments.

Table 13. Salaries, 108 out of 111 U.S. CS Departments								
Faculty rank	Number	Reported minimums Min	Mean	max	Average over all salaries	Reported maximums Min	Mean	Max
Assistant	760 (772)	29000	38895	43000	41945	33100	43852	56425
Associate	530 (535)	28311	42788	57882	47428	36700	51660	66640
Full	662 (675)	34483	51444	72420	63037	46200	74672	125000

Table 14. Salaries, 12 of 12 CS Departments ranked 1-12, U.S. only								
Faculty rank	Number	Reported minimums Min	Mean	Max	Average over all salaries	Reported maximums Min	Mean	Max
Assistant	126 (126)	39000	41048	43000	43148	42800	46286	55000
Associate	73 (73)	28311	44962	55000	49301	45783	53348	61900
Full	143 (143)	34483	54479	71400	70330	73377	92993	125000

Table 15. Salaries, 11 of 12 CS Departments ranked 13-24, U.S. only								
Faculty rank	Number	Reported minimums Min	Mean	Max	Average over all salaries	Reported maximums Min	Mean	Max
Assistant	109 (115)	39000	40674	43000	42987	43000	46388	55492
Associate	65 (68)	35000	45979	52000	51544	48650	57449	66640
Full	82 (90)	48480	53510	61000	65813	66400	84461	97000

Table 16. Salaries, 11 of 12 CS Departments ranked 25-36, U.S. only								
Faculty rank	Number	Reported minimums Min	Mean	Max	Average over all salaries	Reported maximums Min	Mean	Max
Assistant	97 (99)	37100	40499	42000	43307	37100	45849	45109
Associate	54 (55)	34750	45633	51400	51112	44900	53945	61320
Full	68 (72)	46200	54146	61700	66970	75000	87793	122100

Table 17. Salaries, 74 of 87 CS Departments ranked below 36 or unranked, U.S. only								
Faculty rank	Number	Reported minimums Min	Mean	Max	Average over all salaries	Reported maximums Min	Mean	Max
Assistant	428 (432)	29000	38007	43000	41017	33100	42739	56425
Associate	337 (339)	30000	41547	57882	45779	36700	50168	64925
Full	369 (370)	35525	50209	72420	58869	46200	68118	104750

Table 18. Salaries, 17 of 33 CE Departments, U.S. only								
Faculty rank	Number	Reported minimums Min	Mean	Max	Average over all salaries	Reported maximums Min	Mean	Max
Assistant	92 (110)	34946	38381	42500	40603	36146	42647	50000
Associate	103 (128)	35600	41900	48700	47871	39800	49503	56000
Full	150 (184)	31425	47216	74000	54907	50375	67722	89624

Table 19. Salaries, 11 of 12 Canadian CS Departments (Canadian dollars)								
Faculty rank	Number	Reported minimums Min	Mean	Max	Average over all salaries	Reported maximums Min	Mean	Max
Assistant	75 (85)	37674	41620	48000	44958	42518	48241	55300
Associate	91 (95)	38500	48299	53980	54766	50483	60659	69624
Full	91 (107)	49328	58677	68578	69625	67434	81108	97774

Table 20. Salaries, 126 of 133 CS and CE U.S. Departments								
Faculty rank	Number	Reported minimums Min	Mean	Max	Average over all salaries	Reported maximums Min	Mean	Max
Assistant	853 (883)	29000	38823	43000	41793	33100	43684	56425
Associate	640 (670)	28311	46888	57882	46981	36700	51368	66640
Full	815 (862)	31425	50920	74000	61456	46200	73810	125000

Five-year estimates of department growth

The departments were asked to estimate their faculty sizes through 1992-93, given an adequate supply of applicants (the lack of applicants has been a problem in the past). The 145 CS-CE departments would like to grow by 927 (33%) in the next five years. The 123 CS departments would like to grow by 808 (35%); last year the 117 departments expected to grow by 37%.

Last year, the 117 departments reported a desire to grow from 2173 (18.6 per department) in 1986-87 to 2387 (20.4 per department) faculty members by 1987-88. However, the 123 CS departments this year reported growing only to 2325 (18.9 per department). Table 22 indicates that all departments desire substantial growth, but with the most growth expected in the less well-ranked and in the smaller departments.

The right half of Table 22 seems strange — it indicates that, in four of the groups, the average faculty size *decreased* from 1986-87 to 1987-88. This is explained as follows. There were six more departments in 1987-88, and these were rather small. Further, growth in a department from 39 to 41 (say) would move the department into the group 40-49, thus reducing the average department size in each of the groups 30-39 and 40-49.

Table 21. Desired Faculty Growth									
		1987-88	1988-89	1989-90	1990-91	1991-92	1992-93	5-year increase	
CS	Faculty size	2325	2543	2722	2893	3031	3133	808	(35%)
	Average size	18.9	20.7	22.1	23.5	24.6	25.5	6.6	
CS-CE	Faculty size	2806	3055	3269	3460	3616	3733	927	(33%)
	Average size	19.4	21.1	22.5	23.9	25.0	25.7	6.3	

Table 22. Average Desired Five-Year Growth in CS Departments										
	By rank				By department size				
Per department	1-12	12-24	24-36	rest	1-9	10-19	20-29	30-39	40-49
Number of depts. 1987-88	12	12	12	87	8	67	31	12	4
Average dept. size 1986-87	30	25	20	16	8	15	25	34	46
Average dept. size 1987-88	30	26	21	16	7	14	24	34	43
Average dept. size 1992-93	35	32	31	22	11	21	31	39	57
Average five-year increase	5	6	10	8	4	7	7	5	14
Percent growth (projected)	17%	23%	48%	50%	57%	50%	29%	15%	33%

Faculty Losses

As shown in Table 23, the field of computing lost only 28 people through death or retirement, which is about 1% of the total number of faculty; CS lost 18 — the same as last year. This, together with the distribution of the faculty in the three ranks, points out the extreme youth of the field. Of the other CS-CE 188 faculty who left, at least 40% left for other teaching positions, 25% left academia, 12% were visitors who returned to their employer, 2% returned to graduate school, and 8% other. The percentages for CS were very similar: 44% teaching elsewhere, 25% positions outside of academia, 11% were visitors, 3% returned to graduate school, and 8% other. The number of faculty who left the departments (179) is very close to the figure reported last year (174).

Table 23. Faculty Losses

	CS-CE Depts.			CS Depts.		
	w/ Ph.D.	w/ out Ph.D.	Total	w/ Ph.D.	w/ out Ph.D.	Total
Died or retired	23	5	28	14	4	18
Were visitors, returned to employer	25	1	26	19	1	20
Teaching elsewhere	82	5	87	73	5	78
Left for non-academic position	46	7	53	38	6	44
Returned to graduate school	2	3	5	2	3	5
Other	16	1	17	13	1	14
Total	194	22	216	159	20	179

References

[0] An assessment of research-doctorate programs in the United States. National academy Press, 1982.

[1] Gries, D. The 1984-85 Taulbee survey. *CACM 29*, 10 (October 1986), 972-977; and *Computer* 19, 10 (November 186), 69-71.

[2] Gries, D. The 1985-86 Taulbee survey. *CACM 30*, 8 (August 1987), 688-694.

[3] Taulbee, Orrin E. Annual U.S. summaries of Ph.D. production and employment in computer science, 1970-1985. *SIGCSE Bulletin 18*, 3 (September 1986), 2-8, 12.

[4] First Report of the 1987 Annual AMS-MAA Survey. *Notices of the American Mathematical Society* (November 1987).

THE UNIVERSITY CONTEXT

The State-Wide University System

The State of California supports two parallel and occasionally competing University systems. Until a few years ago, the members of one of these systems bore the designation "State Colleges" to distinguish them from the various campuses of the University of California. After some controversy, the State Legislature authorized several of these State College units to call themselves "State Universities" There are now 17 State Universities and 2 State Colleges within the California State University System. In Southern California, the largest of these include California State University, Fullerton; California State University, Long Beach; California State University, Los Angeles; California State University, Northridge; and California State University, San Diego. The estimated state-wide enrollment for all nineteen campuses of the system was 342,776 for Fall 1987. The California State University System is headed by a Board of Trustees which in turn selects the Chancellor of the State University system and the presidents of each of the State Universities. Dr. W. A. Reynolds is the Chancellor at the present time.

The University of California likewise depends on the State of California Legislature for its funding. It is governed by a Board of Regents consisting of twenty-four members, seventeen of whom are appointed to sixteen-year terms by the Governor of California, and seven members ex-officio (including the Governor, the Lieutenant Governor and the Speaker of the Assembly). The Board of Regents prepares the University budget for submission to the State Legislature and rules on all changes in academic programs and all tenure professorial and senior administrative appointments. The Board of Regents also appoints the President of the University of California as well as several Vice Presidents. In 1983 Dr. David P. Gardner became President, succeeding Dr. David Saxon.

The Campuses of the University of California

The University of California maintains nine major centers or campuses, as well as a number of smaller peripheral units. The oldest of these is at Berkeley and began operation at its present location in 1873. The most recent addition to the University system is the Santa Cruz campus which opened officially in Fall 1965 and is still undergoing rapid changes. Within the University of California system, each campus enjoys autonomy. Although some campuses are relatively specialized (for example the San Francisco campus is devoted to medicine and the health sciences), most campuses offer complete programs leading to Bachelor's, Master's and Ph.D. degrees. While it is common for a student to transfer from one campus of the University of California to another after completing the Bachelor's or Master's degree, it is relatively rare to make a change from campus to campus while pursuing a specific degree objective. A list of the campuses of the University of California together with their graduate and undergraduate student enrollment in Fall 1987 follows.

Campus	Undergraduate	Graduate	Health Sciences	Total
Berkeley	22,636	8,634	785	32,055
Davis	15,486	3,484	1,895	20,847
Irvine	12,369	1,695	1,075	15,139
Los Angeles	23,447	8,194	3,794	35,435
Riverside	5,023	1,488	43	6,554
San Diego	13,589	1,690	1,118	16,397
San Francisco			3,681	3,681
Santa Barbara	15,777	2,102		17,879
Santa Cruz	8,512	640		9,152
UC Totals	116,821	27,927	12,391	157,139

The University of California at Los Angeles

UCLA was founded in 1919 as the second branch of the University of California. It moved to its

present Westwood location in 1929. Berkeley and UCLA are the two largest University campuses student enrollment. UCLA is located on 411 acres at the western edge of Los Angeles, approximately 7 miles from the Pacific Ocean. In addition to the beauty of its campus, the uniqueness of its sculpture garden, and the stimulation provided by its arts, sciences, and professional schools, it boasts a research library containing approximately 5.5 million volumes, an increasingly distributed campus computing network with a large computer system at its center, one interactive computation facility called SEASNet in the School of Engineering, another in the Computer Science department, and many other major research facilities.

The chief administrative officer of the UCLA campus is Chancellor Charles E. Young who has held this office since 1968 when he succeeded Chancellor Franklin Murphy. Faculty self-governance is effected through the Academic Senate. Assistant Vice-Chancellors maintain cognizance over key domains such as academic affairs, research, student support and facilities. Dean Victoria Fromkin of the Graduate Division is concerned with graduate study and research.

The academic programs at UCLA are organized around two Colleges and eleven Schools as well as a number of organized research units. The distinction between a College and a School is that a College is obligated to provide undergraduate instruction leading to a Bachelor's degree, while a School may or may not choose to provide undergraduate instruction.

The two UCLA Colleges are the College of Fine Arts and the College of Letters and Sciences. The latter, the larger of the two, is headed by Provost Raymond Orbach and is divided into four Divisions: Humanities, Life Sciences, Physical Sciences and Social Sciences. Each of these divisions contains a number of departments. The Departments of Chemistry, Mathematics, and Physics, whose courses are fundamental to students of engineering and computer science are within the Division of Physical Sciences. There are ten professional schools in addition to the School of Engineering and Applied Science.

The School of Engineering and Applied Science

Engineering began as an independent program at UCLA in 1945 shortly after the late L.M.K. Boelter arrived from Berkeley to head the College of Engineering. In its early years UCLA provided a unified curriculum leading to the degree of Bachelor of Science in Engineering. Graduate instruction commenced a few years later, and by the time of the retirement of Dean Boelter in 1965, the College of Engineering had 1,254 undergraduate and 969 graduate students. After Chauncey Starr became Dean in 1967, the College of Engineering became the School of Engineering and Applied Science to highlight its emphasis on graduate education and was divided into departments. Russell R. O'Neill succeeded as Dean in 1973. Dean O'Neill retired and George L. Turin was appointed Dean effective 1983. From 1983 to 1986 the complete decentralization of SEAS into more classical departments was effected. Presently, A.R. Wazzan is the Dean for the School of Engineering and Applied Science.

The School is subdivided into six academic departments: Chemical Engineering, Civil Engineering, Computer Science, Electrical Engineering, Materials Science and Engineering, and Mechanical, Aerospace and Nuclear Engineering. The following were the approximate number of graduate students enrolled in each of the six departments of the School of Engineering and Applied Science for Fall 1987.

Chemical Engineering	46
Civil Engineering	67
Computer Science	272
Electrical Engineering	295
Materials Science & Engineering	63
Mechanical, Aerospace and Nuclear Engineering	153

Short courses and the evening extension program provide a valuable service to Los Angeles area engineers and scientists wishing to continue their education.

The Faculty: Areas of Interest

In the 1987-88 academic year the full-time faculty with primary appointment in the Computer Science Department included twenty-two Professors, three Associate Professors, five Assistant Professors and one Senior Lecturer with Security of Employment. Two faculty members with primary affiliations in other departments at UCLA contributed substantially to the Departmental teaching and research programs as did six faculty members with visiting or temporary appointments. Their fields of interest by area, brief biographies, publications, and honors and awards follow.

For purposes of specifying Ph.D. major and minor fields as well as to characterize research interests, the Computer Science program at UCLA has been divided into six areas. Each area is represented in the curriculum by a set of undergraduate and graduate courses, and is supervised by a committee of faculty members having research and pedagogic interests in that area. A number of faculty members belong to more than one such area. Listed below are the more important research topics falling under each area together with the associated faculty members.

•Artificial Intelligence

Artificial intelligence: heuristics, machine perception, natural language processing, knowledge-based and expert systems, robotics, graphics and man/machine interfaces.

R. Korf *(area head)*, M. Dyer, M. Flowers, A. Klinger, M. Melkanoff, S. Parker, J. Pearl, J. Skrzypek, J. Vidal.

•Computer Network Modeling and Analysis

Computer Networks and Packet Switching; Distributed Systems; Flow Control; Routing Control; Packet Radio Systems, Local Networks; Computer Scheduling, and Time Sharing; Resource Allocation; Memory Management; Data Communications; Performance Evaluation, Queueing Theory; Network Flow Theory; Mathematical Modeling, Analysis and Optimization of Computer Systems.

L. Kleinrock *(area head)*, J. Carlyle, W. Chu, E. Gafni, M. Gerla, S. Greibach, R. Muntz, D. S. Parker.

•Computer Science Theory

Design and Analysis of Algorithms, Computational Complexity, Models for Program and Data Structures, Models for Parallel Computation and Concurrent Systems, Verification Systems, Automata and Formal Languages, Discrete System Theory.

S. Greibach *(area head)*, D. Cantor, J. Carlyle, E. Gafni, D. Martin, D. S. Parker, J. Pearl

•Computer System Architecture

System Design; Digital Systems; Logic Design; VLSI System Design and Organization; Memory, Arithmetic, Control, Data Transmission and Input/Output System Design; Computer Graphics; High-Performance Computers; Fault-Tolerant Computers, Distributed Systems.

D. Rennels *(area head)*, A. Avizienis, B. Bussell, W. Chu, M. Ercegovac, G. Estrin, M. Gerla, T. Lang, L. Levine, D.S. Parker, G. Popek, V. Tyree, Y. Tamir, C. Viswanathan.

•Scientific Computing

Physical Systems: Modeling and simulation, signal processing, numerical methods, mathematical models, pattern recognition, on-line computations.

Biological Systems: Modeling, neural networks, biocybernetics.

W. Karplus *(area head)*, M. Aoki, J. Carlyle, J. DiStefano, T. Estrin, L. Levine, L. McNamee.

•Software Systems

General and Special-Purpose Languages; Compilers; System Programming; Syntax, Semantics and Pragmatics of Programming Languages; Implementation of Programming Languages and Systems; Distributed Systems; Security and Privacy; Data Bases; Data Structures.

D.S. Parker, *(area head)* R. Bagrodia, D. Berry, A. Cardenas, D. Jefferson, D. Martin, M. Melkanoff, R. Muntz, G. Popek.

COMPUTER SCIENCE DEPARTMENT FACULTY MEMBERS

NAME	LOCATION†	PHONE	LOGIN ID <user@CS.UCLA.EDU
Aoki, Masanao	BH 4731C	825-2360	aoki
Avižienis, Algirdas	BH 4731	825-3028	aviz
Bagrodia, Rajive	BH 3531F	825-0956	rajive
Boehm, Barry	BH 3732	825-2660	boehm
Bussell, Bertram	BH 3732	825-2191	bussell
Campbell, Michael	BH 3531	825-4943	campbell
Cantor, David	MS 6363	825-8188 825-4701*	dgc
Cardenas, Alfonso	BH 3731J	825-7550	cardenas
Carlyle, Jack	BH 3731F	825-8807	jwc
Chu, Wesley	BH 3731C	825-2047	wwc
Dalkey, Norman	BH 3532L	825-2692	dalkey
Dechter, Rina	BH 4731	825-3243	dechter
DiStefano, Joseph	BH 4731K	825-7482	joed
Dyer, Michael	BH 3532E	206-6674	dyer
Ercegovac, Milos	BH 3732C	825-5414	milos
Estrin, Gerald	BH 3713	825-2786	estrin
Estrin, Thelma	UNEX 637	825-4950	testrin
Flowers, Margot	BH 3532G	825-6795	flowers
Gafni, Eliezer	BH 3732D	825-3211	eli
Gerla, Mario	BH 3732H	825-4367	gerla
Glaser, Edward	BH 3732	825-2191	
Gray, Terry	BH 3732	825-7393	gray
Greibach, Sheila	BH 3731G	825-1617	greibach
Inselberg, Alfred	BH 3531	825-1785	inselb
Jefferson, David	BH 3532F	206-6542	jefferso
Karplus, Walter	BH 3732B	825-2929	karplus
Kay, David G.	BH 3531H	825-2695	kay
Kleinrock, Leonard	BH 3732G	825-2543 825-1065	lk
Klinger, Allen	BH 3531C	825-7695	klinger
Korf, Richard	BH 3532H	825-8252	korf
Lang, Tomas	BH 3732E	825-6835	tomas
Levine, Leon	BH 3531E	825-6735	levine
Mak, Patrick	BH 4731J	206-2775	
Martin, David	BH 3532H	825-2091	dmartin
McNamee, Lawrence	BH 3531D	825-7825	mcnamee
Melkanoff, Michel	BH 3532B	825-2212	mam
Muntz, Richard	BH 3731H	825-3546	muntz
Parker, D. Stott	BH 3532E	825-6871	stott
Pearl, Judea	BH 4711	825-3243	judea
Popek, Gerald	BH 3532D	825-6507	popek
Rennels, David	BH 4731D	825-1484	rennel
Simundich, Thomas	BH 4731J	206-2775	tom
Skrzypek, Josef	BH 3532D	825-2381	skrzypek
Tamir, Yuval	BH 4731	206-2852	tamir
Tyree, Vance	BH 4731	825-8137	vance
Vidal, Jacques	BH 3531J	825-2858	vidal
Viswanathan, Chand	BH 7713	825-5214	vis

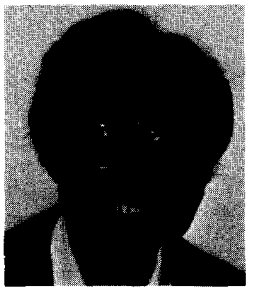
*Messages only.

BH = Boelter Hall; MS = Mathematical Sciences Building

† Message Numbers for each suite are as follows:

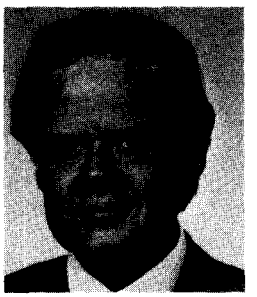
3532 BH - 825-1322; 3531 BH - 825-4943; 3731 BH - 825-2191; 4731 BH - 825-4033

MASANAO AOKI, *Professor*



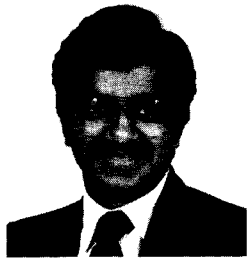
Dr. Aoki was born in Japan. He received the B.S. and M.S. degree in Physics from the University of Tokyo. He earned his Ph.D. in Engineering from UCLA as well as a D.Sc. from the Tokyo Institute of Technology. He has been with UCLA since 1960, except for 1974-76 spent at University of Illinois as Professor of Economics and Electrical Engineering, and for 1981-83 as a Professor of Economics at the Institute of Social and Economic Research, Osaka University. He has held a Social Science Research Council Fellowship and was a Senior Visiting Fellow of the British Science Research council at University of Cambridge, a visiting follow of Clare Hall, and of the Woodrow Wilson School, Princeton University. He is a Fellow of the IEEE, a Fellow of the Econometric Society, and served as President of the Society of Economic Dynamics and Control. His current areas of interest include model construction and selection algorithms for data sources, analysis and control of large or complex dynamic systems, and developments of tools and models to conduct these tasks including various decision support models and systems.

ALGIRDAS AVIZIENIS, *Professor*



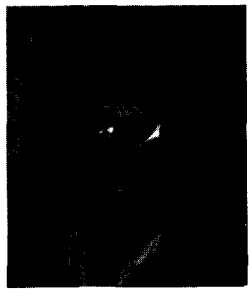
Dr. Avižienis was born in Kaunas, Lithuania, and has resided in the USA since 1950. He received the B.S. (1954), M.S. (1955), and the Ph.D. (1960) in Electrical Engineering from the University of Illinois, where he was associated with the Digital Computer Laboratory as a Research Assistant and Fellow (1956-1960), participating in the design of the ILLIAC II computer. In 1960 he joined the technical staff at the Jet Propulsion Laboratory (JPL) of the California Institute of Technology, where he initiated and directed the JPL-STAR (Self-Testing-And-Repairing) computer research project, for which he received a U.S. Patent in 1970. Dr. Avižienis joined the UCLA Faculty in 1962, concentrating his teaching and research on system architecture, fault-tolerant systems and software, and digital arithmetic. He also remained associated with JPL as an Academic Staff Member, conducting research on fault-tolerant space-craft computers through 1979. Since 1972 Dr. Avižienis has continuously served as the principal investigator of NSF, ONR, NASA, FAA, and industrial research grants on fault-tolerant system design and on software fault tolerance. He served as Chair of the Computer Science Department from 1982 to 1985, and is currently the Director of the Dependable Computing and Fault-Tolerant Systems Laboratory. For his pioneering work in fault-tolerant computing Dr. Avižienis was elected Fellow of IEEE, received the NASA Apollo Achievement Award, the AIAA Information Systems Award, the NASA Exceptional Service Medal, the IEEE Computer Society Technical Achievement Award and the IFIP Silver Core Award. In 1985 he was presented with the honorary *Docteur Honoris Causa* degree by the Institut National Polytechnique in Toulouse, France. In 1980 Dr. Avižienis founded the Working Group 10.4 on "Reliable Computing and Fault-Tolerance" of IFIP, the International Federation for Information Processing, which he chaired through 1985. Since 1984 he serves as the fault tolerance expert on the Technical Advisory Group of the FAA Advanced Automation Program for air traffic control.

RAJIVE BAGRODIA, *Assistant Professor*



Dr. Bagrodia was born in Calcutta, India. He received the Bachelor of Technology degree in Electrical Engineering from the Indian Institute of Technology, Bombay in 1981. He was awrded the M.A. and Ph.D degrees in Computer Science by the University of Texas at Austin in 1983 and 1987 respectively. Dr. Bagrodia joined the Computer Science Department at UCLA in July, 1987. His current areas of research include distributed algorithms, concurrent programming languages, simulation, performance evaluation of distributed systems and methodologies for the design of integrated systems.

DANIEL M. BERRY, *Professor*

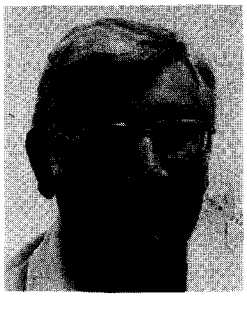


Dr. Berry was born in Cleveland Heights, Ohio. He received his B.S. in Mathematics from Rensselaer Polytechnic Institute in 1969. He went to graduate school at Brown University, during which time he worked at General Electric R & D Center in Schenectady, New York and taught at the Hebrew University in Jerusalem, Israel. He joined the UCLA Computer Science Department faculty as an Acting Assistant Professor in September 1972. He completed his Ph.D. thesis in September 1973 for his degree in Applied Mathematics/Computer Science from Brown. Shortly thereafter he was promoted to Assistant Professor. He worked his way through the ranks becoming an Associate Professor in 1977 and a Professor in 1983. During the 1979-79 academic year he spent a sabbatical at the Hebrew University in Jerusalem, Israel and at the Weizmann Institute in Rehovot, Israel. He has spent extended research visits at various places including Pontifícia Universidade Católica in Rio de Janeiro, Brazil and Politecnico di Milano in Milano, Italy. For several years, he was an active member of the program committee of the annual Conferencia Internacional de Ciencias de la Computacion in Santiago, Chile.

BARRY BOEHM, *Adjunct Professor*

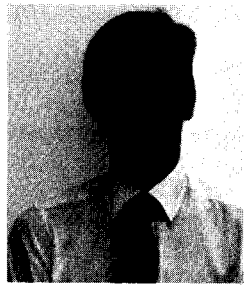
Dr. Boehm was born in Santa Monica, California. He received his B.A. in Mathematics from Harvard in 1957 and his M.A. and Ph.D. in Mathematics from UCLA in 1961 and 1964, respectively. He was Head of the Information Sciences Department at The Rand Corporation until 1973, when he joined TRW; he is currently Chief Scientist of TRW's Defense Systems Group. He has been an Adjunct Professor of Computer Science at UCLA since 1983. His research interests center around software engineering, particularly in the areas of software development environments, requirements methodology, and software metrics. He serves on the Governing Board of the IEEE Computer Society, the Defense Science Board Task Force on Software, the National Academic Advisory Board of the Wang Institute, the Board of Directors of the Rocky Mountain Institue of Software Engineering, and the editorial boards of several journals. His most recent book, *Software Engineering Economics*, was published by Prentice-Hall in 1981.

BERTRAM BUSSELL, *Professor Emeritus*



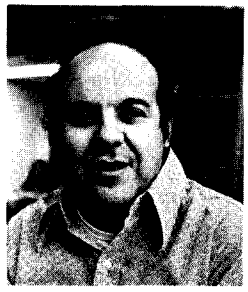
Dr. Bussell was born in New York and was educated entirely at UCLA, receiving the Bachelor degree in 1950, the Master's degree in 1952, and the Ph.D. in 1962. He joined the UCLA Faculty in 1963 and is concentrating his teaching and research in the computer architecture area. His special interests include computer graphics, graph models of computation and networks of microprocessors.

MICHAEL CAMPBELL, *Lecturer*



Dr. Campbell received his B.S. in Mathematics from UC Riverside in 1980; his M.S. in Computer Science from UCLA in 1982; and his Ph.D. in Computer Science from UCLA in 1986. Currently, he teaches the undergraduate course in Formal Language and Automata Theory. His research interests include: computability and complexity theory, parallel algorithms, software development tools/environments for distributed systems, artificial neural networks, and machine learning.

DAVID CANTOR, *Professor, Department of Mathematics*



Dr. Cantor was born in London, England and received his B.S. in Physics from California Institute of Technology, and his Ph.D. from UCLA in Mathematics in 1960. He taught at Princeton University and the University of Washington before coming to the UCLA Mathematics Department. His research interests include algorithms, combinatorics (especially computer methods) and number theory (especially diophantine approximation).

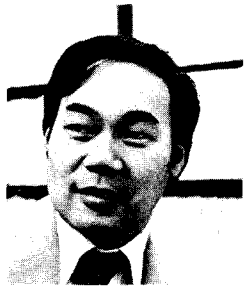
ALFONSO F. CARDENAS, *Professor*

Dr. Cardenas was born in Aguascalientes, Mexico and received the B.S.E.E. from San Diego State University in 1964 and the M.S. and Ph.D. degrees from UCLA in 1966 and 1969, respectively. He joined the UCLA Faculty in 1969. He was Visiting Scientist/Consultant at IBM in 1972, 1973, and 1974. His research interests include data management and data base systems, management information systems, automatic programming, programming languages and image processing. He is the author of the book *Data Base Management Systems,* 1979 (1984, 2nd edition). He is the current president of the Very Large Data Base Endowment. He serves on the editorial board of the *Information Systems Journal* and on the board of directors of two companies.

JACK CARLYLE, *Professor*

Dr. Carlyle was born in Cordova, Alaska; he received the B.A. (Mathematics) in 1954 and the M.S. (Electrical Engineering) in 1957 from the University of Washington, and the M.A. (Statistics) and Ph.D. (Electrical Engineering) in 1961 from the University of California, Berkeley. He was Assistant Professor of Electrical Engineering at Princeton in 1961-62, then joined the UCLA faculty in 1963 as a member of the Information Systems Division, the predecessor of our present Computer Science Department. Professor Carlyle's areas of research interest include data communication and computer methodology, discrete-state and stochastic systems, algorithms and complexity.

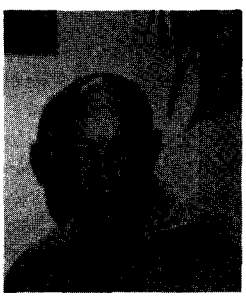
WESLEY CHU, *Professor*



Dr. Chu was born in Shanghai, China and received the B.S. and M.S. degrees from the University of Michigan. He earned the Ph.D. from Stanford in 1966. He has worked on switching circuits and computer systems at General Electric (now Honeywell) and at IBM and did his research in computer communications at Bell Labs, 1966-1969. He joined the UCLA Faculty in 1969 and is centering his research in computer system modeling and analysis, distributed processing, and

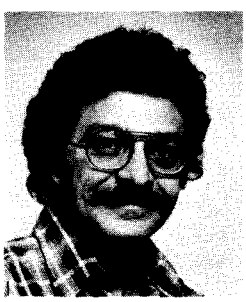
distributed data base systems. He was the past ACM SIGCOMM Chairman (1973-1977), Associate Editor of the IEEE Transaction in Computers on distributed computing and computer networking (1978-82), and Guest Editor of the IEEE Proceedings on Distributed Database Systems (May, 1987). In 1983 he received the IEEE Meritorius Service Award for his work as editor. He also served on the editorial board of several journals. He is the Principal Investigator of several projects on Distributed Processing and Distributed Data Base Systems. He has published over 75 technical articles and edited three textbooks entitled Advances in Computer Communications and Networking, Centralized and Distributed Data Base System (co-edited with P.P. Chen), Distributed Systems: Distributed Processing Systems (Vol. I), Distributed Database Systems, (Vol. II), and is a Fellow of the IEEE.

NORMAN DALKEY, *Adjunct Professor*



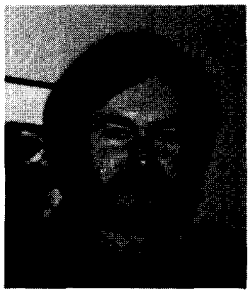
Dr. Dalkey received a Ph.D. in Philosophy from UCLA in 1942. During World War II he taught Meteorology at UCLA and did research as a physicist at the Radiation Laboratory at Berkeley. He joined the Rand Corporation in 1948 as a Senior Mathematician where he did work on the Theory of Games, computer simulation, and was co-developer of the Delphi technique for group judgement. He became associated with the Engineering Systems Department at UCLA as Adjunct Professor in 1972, and moved to Computer Science in 1983. He is co-director of the Cognitive Systems Laboratory. His research on Group Judgement and Information Systems has been supported by ARPA and NSF.

JOSEPH J. DISTEFANO, III, *Professor of Computer Science & Medicine*



Professor DiStefano was born in Brooklyn, New York in 1938. He received his B.E.E. in Electrical Engineering from the City College of New York in 1961. His Master's degree and Ph.D. degree were awarded from UCLA in 1964 and 1966 in the fields of Control Systems and Biocybernetics with minor field studies in Physiology, Computer Methodology and Applied Mathematics. He joined the UCLA School of Engineering faculty in 1966 and the UCLA Department of Medicine faculty in 1970. He has been a Visiting Professor and NATO Postdoctoral Fellow in Endocrinology at the University of Rome, Italy (1967-68); a Visiting Professor at the Department of Physiology, University of Laval in Quebec (Fall 1971); Visiting Professor, Department of Nuclear Medicine and Division of Automatic Control, Lund Institute of Technology, Lund, Sweden (January-September 1976); Senior Fullbright Scholar at the National Research Council of Italy (CNR) in Padova, Italy (Spring 1981); and Visiting Professor in the System Science Department of the City University, London, England (April-August 1984). He is the chairman of the Cybernetics Interdepartmental Program in UCLA, founder and Head of the Biocybernetics Laboratory, and Editor of the Modeling Methodology Forum of the American Journals of Physiology and Journal of Applied Physiology. His research interests are in Biocybernetics, with emphasis on modeling theory, expert systems for modeling, optimal experiment design, endocrine and metabolic systems, interactive simulation and graphics. He is currently directing projects in these areas.

MICHAEL DYER, *Associate Professor*



Dr. Dyer was born in Washington D.C. and received an AB in English from Dartmouth College in 1970, an MA in Anthropology from Temple University, and an MS in Computer Science from the University of Kansas. From 1978 to 1982 he was a graduate fellow and research assistant at the Yale Artificial Intelligence Lab. After receiving his Ph.D. from Yale, he worked at Cognitive Systems, Inc. in New Haven, CT as a Senior Research Scientist. He is a recipient of a 1983 IBM Faculty Development Award, 1984 TRW Excellence in Teaching Award, and author of a book titled "In-Depth Understanding", MIT Press 1983. His research interests include: natural language comprehension and modelling human cognitive skills such as: reasoning, planning, learning, and creativity. He is currently director of the CS department Artificial Intelligence Laboratory.

MILOS D. ERCEGOVAC, *Professor*

Dr. Ercegovac was born in Belgrade, Yugoslavia. He received his diploma in Electrical Engineering from the University of Belgrade in 1965 and M.S. and Ph.D. degrees in Computer Science from the University of Illinois, Urbana-Champaign, in 1972 and 1975, respectively. He was associated with Brown & Boveri Laboratories, Baden, Switzerland ('65), with the Institute for Automation and Telecommunications "Mihailo Pupin" in Belgrade ('66-'70), and with the Department of Computer Science of the University of Illinois ('70-'75). His research and teaching interests include Fast Arithmetic and Hardware-Oriented Algorithms, High-Speed Computer Architectures, Logic Design, and Functional (applicative) Languages and Data-Driven Machines.

GERALD ESTRIN, *Professor*

Dr. Estrin was born in New York and educated at the University of Wisconsin where he received the Bachelor's, Master's and Ph.D. degrees in 1948, 1949, and 1951, respectively. He served as research engineer in Von Neumann's group at the Institute for Advanced Study, Princeton, where he participated in the design of one of the earliest large digital computers. In 1954 he served as the Director of the Electronic Computer Project at Israel's Weizmann Institute of Science where he led the development of WEIZAC, the first large-scale electronic computer outside of the United States or Western Europe. He is an IEEE Fellow, a Guggenheim Fellow, and a member of the Board of Governors of the Weizmann Institute of Science, Israel. Dr. Estrin joined the UCLA Faculty in 1956 and in 1979 received University-wide recognition when the Regents approved his appointment as an Above Scale Professor. He is leading research projects on design methodology and computer based tools for modeling, measurement and synthesis of concurrent computer systems. Dr. Estrin's most current research interests are concerned with environments to support collaborative design by small teams. He served as Chair of the UCLA Computer Science Department from 1979-1982 and has served as the current Chair since September 1985.

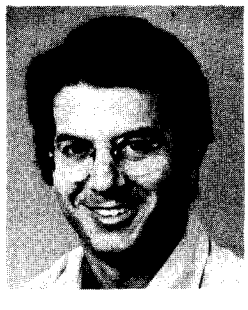
THELMA ESTRIN, *Professor-In-Residence*

Dr. Estrin received the B.S., M.S., and Ph.D. degrees in Electrical Engineering from the University of Wisconsin in 1948, 1949, and 1951 respectively. Upon completion of her Ph.D. studies she was employed as an Electrical Engineer in the EEG Laboratory of the Columbia Prebysterian Neurological Institute. In the mid-fifties, she was a member of the engineering group which built the first electronic computer in the Middle East at the Weizman Institute of Science, Israel. In 1961, she joined the staff of the Brain Research Institute's Data Processing Labortory, of which Dr. Estrin was Director from 1970 to 1980. In 1980 Dr. Estrin joined the School of Engineering and Applied Science as a Professor in Residence in the Computer Science Department with research interest in computer methodology for biomedical systems. In July 1984 Dr. Estrin became Director of the UCLA Extension Department of Engineering and and Science and Assistant Dean for Continuing Education in the School of Engineering and Applied Science. She was recruited for the position through a nationwide search while on leave from UCLA serving as Director of the Electrical, Computer, and Systems Engineering Division of the National Science Foundation. Dr. Estrin is a consultant to industry and government. She is also active in professional societies and was the 1982 Executive Vice President of IEEE. Her honors include Fellow of IEEE and AAAS, Distinguished Service Citation, University of Wisconsin; 1981 Achievement Award of the Society of Women Engineers; and IEEE Centennial Medal.

MARGOT FLOWERS, *Assistant Professor*

Margot Flowers was born in Los Angeles and attended Wellesley College in Boston as a National Merit Scholar. She received a BA in Anthropology at Temple University in 1973 and an MS in Computer Science at the University of Kansas in 1978. From 1978 to 1982 she was a graduate fellow and research assistant at Yale University's Artificial Intelligence Laboratory. In 1981 she served on the program committee of the 7th International Joint Conference on Artificial Intelligence. She expects to receive her Ph.D. from Yale shortly. Her research interests include: computer models of human political and economic knowledge, human reasoning and argumentation, scientific and legal reasoning and belief systems.

ELIEZER GAFNI, *Associate Professor*



Dr. Gafni receved his B.Sc. in 1972 from the Technion Israel, his Ms.C in 1979 from the university of Illinois Urbana-Champaign, and his Ph.D. in 1982 from M.I.T., all in Electrical Engineering. From 1972 to 1977 he served in the Israeli navy as a technical officer. Dr. Gafni's research is in the areas of computer networks and distributed systems with special emphasis on mathematical and algorithmic optimization. Dr. Gafni is the recipient of I.B.M Faculty Development Award 1984, and N.S.F Presidential Young Investigator Award 1985.

MARIO GERLA, *Professor*

Dr. Gerla was born in Milan, Italy. He received a graduate degree in engineering from the Politecnico di Milano, in 1966, and the M.S. and Ph.D. degree in engineering from UCLA in 1970 and 1973, respectively. From 1973 to 1976 he was with Network Analysis Corporation, New York, where he was involved in several computer network design projects for both government and industry. He joined the faculty of the UCLA Computer Science Department in 1977. His research interests cover the performance evaluation, design and control of distributed computer communication systems and networks.

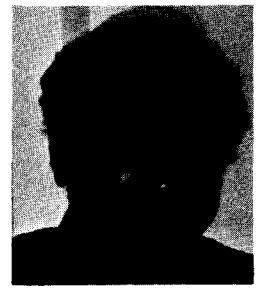
EDWARD L. GLASER, *Visiting Professor*

Edward Glaser received the A.B. degree from Dartmouth College in 1951. He has worked for IBM, where he was a member of the Planning Group on Large Size Computer and participated in the design of 15 computer systems. He then worked for Burroughs Corporation, where he established one of the first Computer Science Research Departments in industry and participated or headed several design projects; M.I.T., where he was on the faculty and was the principal architect of the MULTICS project; Case Western Reserve University, as director of the Andrew R. Jennings Computer Center; and System Development Corporation, as Chief Technical Officer of the Commercial Products Division. He was Director of Advanced Computer Systems Technology Memory Products Division of Ampex Corporation in El Segundo, California from 1979 through 1981. He is currently Chairman of the Board and Chief Technical Officer of Marcus Information Systems, Los Angeles, California, which he founded in 1982. Mr. Glaser is a member of the Board of Trustees of The Seeing Eye in Morristown, N.J., and the Sensory Aids Foundation in Palo Alto, California, as well as an IEEE Fellow. He has been a member of the National Academy of Engineering since 1977. His honors include membership in Phi Beta Kappa, being named Computer Man of the Year for 1974 by the Data Processing Management Association (USA), and receiving an honorary D.Sc. degree from Heriot-Watt University, Edinburgh, Scotland, in 1980.

SHEILA GREIBACH, *Professor*

Dr. Greibach was born in New York City and received the A.B. degree from Radcliffe College in linguistics and applied mathematics in 1960. She received the A.M. degree from Radcliffe in 1962 and the Ph.D. from Harvard University in 1963, working on the machine translation project at the Harvard Computation Laboratory. She served on the faculty of the Harvard Division of Engineering and Applied Physics for six years, spending several summers in Santa Monica at the System Development Corporation, and joined the UCLA faculty in 1969. She is currently Vice Chair for Graduate Studies and has served as Vice Chair for research. She has published over 60 papers in refereed journals and is the Editor-In-Chief of Computer Science Theory (formerly Mathematical Systems Theory). Her interests include algorithms and computational complexity, the complexity of parallel and distributed computation, Petri nets and models for asynchronous computing, formal languages and automata theory program schemes and semantics, and computability.

ALFRED INSELBERG, *Adjunct Professor*

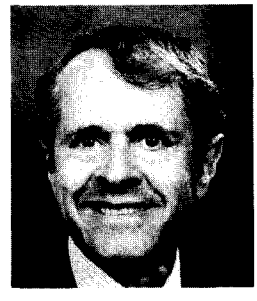


Dr. Inselberg received a B.Sc. in Engineering in 1958 and a Ph.D. in Applied Mathematics in 1965 from the University of Illinois, where he was a member of the Biological Computer Laboratory (BCL) - a cybernetics group - in the Electrical Engineering Department. He was Research Assistant Professor at BLC from June of 1965 to September of 1966. He then joined the IBM Los Angeles Research Scientific Center where he is to-date. In 1985, Dr. Inselberg was appointed IBM Corporate Senior Technical Staff Member. He has lectured at various instutions in the United States and abroad: Technion, Israel; Ben-Gurion University, Israel; University of Southern California; and at the University of California at Los Angeles. Currently Dr. Inselberg holds two Adjunct appointments, one at the USC Computer Science department and one at the UCLA Computer Science department. His research interests include multi-dimensional graphics, computational geometry and applications to intelligent instrumentation and process control. Dr. Inselberg has developed a methodology for multi-dimensional graphics which has been applied to routing with collision avoidance and early conflict detection with global resolution in air traffic control.

DAVID JEFFERSON, *Assistant Professor*

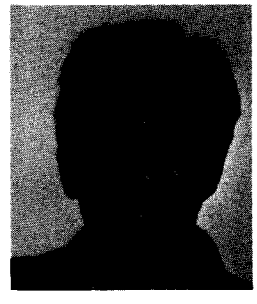
Dr. Jefferson's research centers on the area of distributed systems. His major current interests are in Fault-Tolerance, Dynamic Load Management, and Large-Scale Distributed Simulations. He is one of the developers of the Time Warp mechanism for distributed simulation, which is currently being implemented on the Caltech Hypercube computer under his direction by a team at the Jet Propulsion Laboratory. Dr. Jefferson received a B.S. in mathematics from Yale University in 1970, and a Ph.D. in Computer Science from Carnegie-Mellon University in 1980. He served on the Computer Science faculty at U.S.C. from 1980 to 1984.

WALTER J. KARPLUS, *Professor*



Dr. Karplus was born in Vienna, Austria and received the B.E.E. from Cornell University in 1949 and the M.S. and Ph.D. from the University of California in 1951 and 1955, respectively. He joined the UCLA Faculty in 1955 and was Chairman of the Computer Science Department from 1972 until 1979. Dr. Karplus' research interests include computer methodology, digital simulation and on-line computation. Most recently his research has emphasized the use of multiproccessors and array processors for high-speed real-time simulation and signal processing. He has been National Director of the Society for Computer Simulation since 1973, and was President from 1982 to 1984. He is also a member of the Board of Directors of the American Federation of Information Processing Societies.

DAVID G. KAY, *Lecturer*



David G. Kay was born in Los Angeles, California. He received his B.A. *magna cum laude* in Linguistics from UCLA in 1973 and his Juris Doctor from Loyola Law School (Los Angeles) in 1976. He passed the 1976 bar exam and was admitted to law practice in California. In 1981, he received an M.S. in Computer Science from UCLA under Professor Gerald Popek, with a thesis entitled *Political Regulation of Electronic Funds Transfer Systems.* Mr. Kay joined the

faculty of the UCLA Computer Science Department in 1981; there he has primary responsibility for first-year undergraduate education in computer science, and also offers a graduate seminar on computers and the law. He has presented courses on three continents, to a wide variety of audiences ranging from incoming freshmen with low admissions test scores to senior professors of Engineering. He is the author of the Pascal programming textbook *Programming for People/Pascal* (Mayfield Publishing Co., 1985), and is currently collaborating with Prof. Gary Mar on a textbook on critical reasoning and with Alex Quilici on a Lisp-based data structures text. Mr. Kay is a member of ACM, ABA, the Computer Law Association, and Phi Beta Kappa. His current academic interests include computer law, software engineering, ethics in computing, and computer science education.

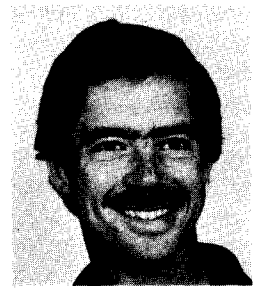
LEONARD KLEINROCK, *Professor*

Dr. Leonard Kleinrock is a professor of Computer Science at the University of California, Los Angeles. He received his B.S. degree in Electrical Engineering from the City College of New York in 1957 (evening session) and his M.S.E.E. and Ph.D.E.E. degrees from the Massachusetts Institue of Technology in 1959 and 1963 respectively. While at M.I.T. he worked at the Research Laboratory in advanced technology. He joined U.C.L.A. in 1963. His research interests focus on local area networks, computer networks, performance evaluation and distributed systems. He has had over 150 papers published and is the author of five books --*Communication Nets: Stochastic Message Flow and Delay, 1964; Queueing Systems, Volume I: Theory, 1975; Queueing Systems, Volume II: Computer Applications, 1976; Solutions Manual for Queueing Systems, Volume I, 1982,* and most recently, *Solutions Manual for Queueing Systems, Volume II, 1986.* Dr. Kleinrock is co-director of the U.C.L.A. Computer Science Department Center for Experimental Computer Science and is a well-known lecturer in the computer industry. He is the principal investigator for the DARPA Advanced Teleprocessing Systems contract at U.C.L.A. Dr. Kleinrock is a member of the National Academy of Engineering, is a Guggenheim Fellow, an IEEE Fellow, a member of the IBM Science Advisory Committee, and in 1986, he became a member of the Computer Science and Technology Board of the National Research Council. He has received numerous best paper and teaching awards, including the ICC 1978 Prize Winning Paper Award, the 1976 Lanchester Prize for outstanding work in Operations Research, and the Communications Society 1975 Leonard G. Abraham Prize Paper Award. In 1982, as well as having been selected to receive the C.C.N.Y. Townsend Harris Medal, he was co-winnr of the L. N. Ericsson Prize, presented by His Majesty King Carl Gustaf of Sweden, for his outstanding contribution in packet switching technology. In July of 1986, Dr. Kleinrock received the 12th Marconi International Fellowship Award, presented by His Royal Highness Prince Albert, brother of King Baudoin of Belgium, for his pioneering work in the field of computer networks. Dr. Kleinrock is also founder and president of Technology Transfer Institute, a computer/communications seminar and consulting organization located in Santa Monica, California.

ALLEN KLINGER, *Professor*

Dr. Klinger was born in New York City. He received the degrees B.E.E. from The Cooper Union, 1957, M.S. from California Institute of Technology, 1958, and Ph.D. from University of California, Berkeley, 1966. His research interests include Computer Vision, Pattern Analysis and Machine Intelligence, and Human-Computer Interaction. He is a Fellow of the Institute of Electrical and Electronics Engineers "for contributions to image analysis by means of computers" and a member of the American Association for Artificial Intelligence, the Pattern Recognition Society, and the IEEE Computer Society. Professor Klinger co-edited the books *Data Structures, Computer Graphics, and Pattern Recognition* and *Structured Computer Vision.*

RICHARD KORF, *Assistant Professor*

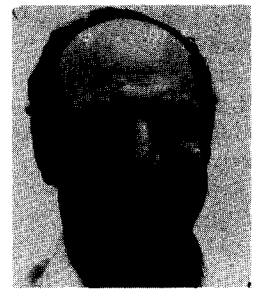


Richard Korf was born in Geneva, Switzerland, in 1956. He received the B.S. in Electrical Engineering and Computer Science from M.I.T. in 1977, and the M.S. in 1980 and Ph.D. in 1983 in computer science from Carnegie-Mellon University. From 1983 to 1985, he served on the faculty of the computer science department at Columbia University, and in 1984 was appointed the first Herbert M. Singer Assistant Professor for his outstanding teaching and contributions to education. Dr. Korf joined the UCLA faculty in 1985. He is the author of *Learning to Solve Problems by Searching for Macro-Operators.* He received an IBM Faculty Development Award in 1985, and an NSF Presidential Young Investigator Award in 1986. He is currently an associate editor of the IEEE journal, *Transactions on Pattern Analysis and Machine Intelligence.* His research interests are in the areas of problem solving, heuristic search, planning, and parallel processing in artificial intelligence.

TOMAS LANG, *Adjunct Associate Professor*

Dr. Lang received his Electrical Engineering degree from the University of Chile in 1963, his Master's degree from the University of California, Berkeley in 1965 and his Ph.D. from Stanford University in 1974. He served as Professor of Electrical Engineering at the University of Chile from 1964 to 1973 and joined the UCLA faculty in 1974. From 1978 to 1981 he was Professor of Computer Science at the Universidad Politecnica de Barcelona. His research and teaching interests are computer architecture and design. At present, his research specializes in digital arithmetic, special-purpose processors for matrix computations, and interconnection networks for parallel computers.

LEON LEVINE, *Senior Lecturer*

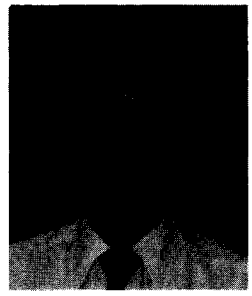


Mr. Levine was born in Bridgeport, Connecticut and received the Bachelor's and Master's degrees at MIT in 1947 and 1949, respectively. Mr. Levine has twenty years of industrial experience, including work on major computer projects at Hughes Aircraft and Scientific Data Systems. He joined the UCLA Faculty in 1967, and is interested in research dealing with simulation and modeling of dynamic systems and design of simulation languages.

LAWRENCE P. McNAMEE, *Professor*

Dr. McNamee was born in Pittsburgh, Pennsylvania and received the B.S.E.E., M.S.E.E. and Ph.D. from the University of Pittsburgh in 1956, 1958, and 1964, respectively. After serving in industrial positions at Westinghouse Electric Corporation and Jones and Laughlin, he joined the UCLA Faculty in 1966. His current research interests include discrete simulation, digital filtering, computer-aided design, microprocessor applications and virtual memory design.

PATRICK H. MAK, *Lecturer*

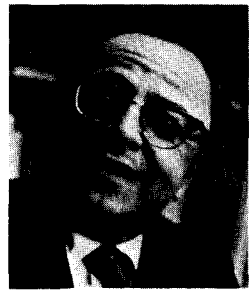


Dr. Mak was born in Hong Kong. He received the B.S. and M.S. both in 1973, and the Ph.D. in Biocybernetics in 1976, all from UCLA. Between 1976 and 1978 he worked at the Garrett Corporation on Energy regeneration technology. Between 1978 and 1980, he was with the Jet Propulsion Laboratory working on interplanetary space programs. In 1981 he joined The Aerospace Corporation where he is now manager of the Advanced Control Applications Section. His research interests include spacecraft attitude determination and control, precision sensor pointing, algorithms for spacecraft autonomous operation, optimal drug delivery system, and applications of control systems methodology to biomedical systems. Dr. Mak has been affiliated with the UCLA faculty since 1976, teaching courses in control systems and biocybernetics.

DAVID F. MARTIN, *Professor*

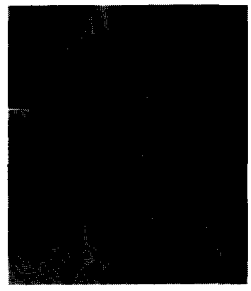
Dr. Martin received the B.S., M.S., and Ph.D. degrees in Engineering from UCLA in 1960, 1962, and 1966. After serving as Assistant Professor of Electrical Engineering at MIT, he returned to UCLA in 1967 to participate in the establishment of the UCLA Computer Science Department and curriculum. His research specialties are the semantics of programming languages and the correctness of programming language implementations. Dr. Martin has twice received the Distinguished Teaching Award from the Engineering Graduate Student Association. In 1985 he received the Distinguished Teaching Award of the Academic Senate.

MICHEL A. MELKANOFF, *Professor*



Dr. Melkanoff was born in Russia and received the B.S. in Aeronautical Engineering from New York University in 1943. He received the M.S. and Ph.D. in Physics at UCLA in 1950 and 1955 respectively. After several years as chief programmer of the UCLA Campus Computing Facility, he joined the faculty in 1962. He was the first Chairman of the Computer Science Department (1969-1972). He was appointed director of the Manufacturing Engineering Program in 1981. In 1987 he was appointed Co-Director for the UCLA/USC Center for Automated Manufacturing Research (CMAR) and to the Steering Committee of IMAR (Institute for Manufacturing and Automated Research). Research interests center on programming languages, management information systems, data bases, data models, CAD/CAM, computer-integrated manufacturing, artificial intelligence in manufacturing, NC machinery, robotics and simulation.

RICHARD R. MUNTZ, *Professor*



Dr. Muntz was born in Jersey City, New Jersey and received the B.E.E. from Pratt Institute in 1963. He was awarded the M.E.E. by New York University in 1966 and the Ph.D. in Electrical Engineering by Princeton University in 1969. He joined the UCLA Faculty in 1969 and has research interests which include Modeling and Analysis of Computer Systems, Operating Systems, and Data Base Systems.

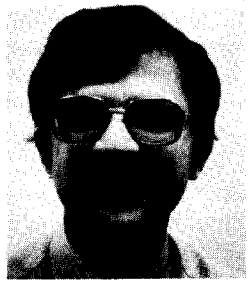
D. STOTT PARKER, *Associate Professor*

D. Stott Parker has been with the UCLA Computer Science Department since January 1979. He received the A.B. degree in Mathematics from Princeton University in 1974, and the Ph.D. in Computer Science in 1978 from the University of Illinois at Urbana-Champaign. Before arriving at UCLA he spent a period of postdoctoral research at the University of Grenoble in France. During the period 1984-1985 he was on a partial leave at Silogic, Inc., a Los Angeles startup company, working as director of research and development of knowledge base management systems. He designed and developed several systems: a Prolog-based KBMS, its interface to relational DBMS, an integrated expert systems development environment, and a window manager for these systems. Professor Parker's main research interests consist in logic programming, knowledge-based systems, database systems, and algorithms. Currently he is engaged in work on development of an advanced computer system performance modeling environment at UCLA, with focus on knowledge representation methods.

JUDEA PEARL, *Professor*

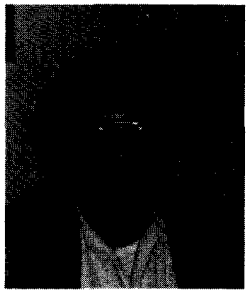
Dr. Pearl was born in Tel-Aviv, Israel. He received the B.S. degree in Electrical Engineering from Technion-Israel Institute of Technology, Haifa, Israel, in 1960; the M.S. degree in physics from Rutgers University, New Brunswick, New Jersey, in 1965; and the Ph.D. degree in Electrical Engineering from the Polytechnic Institute of Brooklyn, Brooklyn. NY in 1965. Before coming to UCLA, he worked at RCA Laboratories, Princeton, New Jersey, on super-conductive parametric and storage devices and at Electronic Memories, Inc., Hawthorne, California, on advanced memory devices. He serves on the editorial boards of AI Journal, Journal of Cybernetics and Systems, and Journal of Intelligent Systems, has authored *Heuristics* (Addison-Wesley, 1984) and has edited *Search and Heuristics* (North-Holland, 1983). He has published over 80 papers in his fields of interest which, currently, include artificial intelligence (especially knowledge-based systems, learning of heuristics, and evidential reasoning), decision aiding systems, and complexity of algorithms.

GERALD POPEK, *Professor*



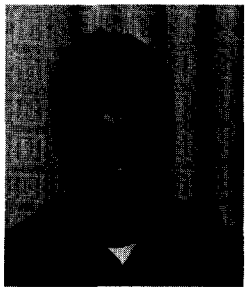
Dr. Popek was born in Passaic, New Jersey and received the B.S. in Nuclear Engineering with honors from New York University in 1968. His Master's and Ph.D degrees were earned at Harvard University in 1970 and 1973 respectively, and he joined the UCLA faculty in 1972. He has research interests centering on privacy and security in computer information systems, the architecture of distributed systems, especially local network based designs, and high-performance parallel expert database systems. He is currently participating in a research project focused on software architectures for high bandwidth, distributed systems and parallel databases and distributed systems.

DAVID A. RENNELS, *Associate Professor*



Dr. Rennels was born in Terre Haute, Indiana in 1942. He received the BSEE degree from Rose Hulman Institute of Technology, Terre Haute, Indiana, in 1964, the MSEE from Caltech in 1965, and the PhD in Computer Science from UCLA in 1973. In 1966 he joined the Spacecraft Computers Section of the Jet Propulsion Laboratory (JPL) of the California Institute of Technology and participated in the design and experimental validation of the JPL-STAR (Self-Testing and Repairing) Computer. In 1976 he initiated the Fault-Tolerant Building Block Computer project at JPL. This program developed an architecture which uses a small number of standard LSI-implementable building block circuits along with existing microprocessors and RAMS to construct fault-tolerant distributed computer systems for spacecraft. Dr. Rennels is currently employed at UCLA where he has been a principal investigator of several research projects sponsored by the Aerospace Corporation, NSF, Hughes Aircraft, and TRW. He has served as a consultant to a number of private companies in the areas of fault-tolerant computer architecture. He is currently involved in the definition and analysis of a very high speed building block architectures for fault-tolerant applications. He is a member of the IFIP Working Group 10.4 on Reliable Computing and Fault-Tolerance, and is currently chairman of the IEEE Computer Society Technical Committee on Fault-Tolerant Computing.

TOM SIMUNDICH, *Lecturer*



Dr. Simundich has held various research and teaching positions at UCLA since 1970. He received the Ph.D. in Computer Science from UCLA in 1975. He currently divides his time between teaching at UCLA and consulting to industry, in particular, for Hughes Aircraft and Rockwell International. His industrial specialties are multitarget tracking, precision pointing mechanisms and servo systems. His academic interests are numerical analysis and system identification.

JOSEF SKRZYPEK, *Assistant Professor*

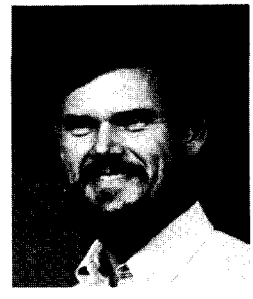
Dr. Skrzypek, began his education in Computer Science at the Wroclaw Polytechnic Institute in 1965. He graduated Magna Cum Laude with BSEE from WNE College in Massachusetts in 1971. His MS (74), and Ph.D. (79) degrees in EECS are from the University of California at Berkeley, where he was supported by the predoctoral NIH Fellowship. His research interests are in the areas of vision, neuronal architectures, artificial intelligence and computational neuroscience. Between 1978 and 1980 he was associated with Electronics Research Laboratories at UCB as a research scientist, where he worked on neuronal architectures for early vision. In 1980 he was awarded NIH Fellowship to continue his studies of connectionism in visual functions at New York University Medical School. Between 1981 and 1982 he served as a consultant and Research Engineer for Analogic Co, developing "real-time" imaging systems for digital angiography. In 1983 he joined the faculty of Northeastern University and in 1984 he was appointed to the Analog Devices Inc. Assistant Professor Chair in Electrical and Computer Engineering. At the same time he received the NSF Research Initiation Award to study connectionist architectures for machine vision. He founded the Vision and Robotics Laboratory and in 1985 was promoted to Associate Professor. Dr. Skrzypek joined the UCLA Faculty in 1986. He is a member of the Computer Science Department and also a principal scientist at the CRUMP Institute of

Medical Engineering. He is the founder and head of the Machine Perception Laboratory at CSD where he is currently directing projects in color and lightness constancy, scheme based PC vision station, intelligent vision sensor, connectionist architecture for multisensory integration and representation of visual knowledge. The research is supported by private and government funds, including NSF, IBM, Analog Devices Inc., ARCO, UCLA, DARPA, ONR and Hughes.

YUVAL TAMIR, *Assistant Professor*

Dr. Tamir was born in Tel-Aviv Israel. He received the BSEE degree from the University of Iowa, Iowa City, in 1979 and the MS and Ph.D. degrees in Electrical Engineering and Computer Sciences from the University of California, Berkeley, in 1981 and 1985, respectively. From 1976 to 1979 he was a research assistant at the University of Iowa Computer Center. From 1979 to 1985 he was a reseach assistant at the Computer Science Division, UC Berkeley. Dr. Tamir joined the UCLA Computer Science Department in July 1985. His research interests include computer systems architecture, architectures for VLSI implementation, fault-tolerant computing, parallel processing, interprocessor communication in multicomputer systems, implementation of self-checking VLSI modules, error recovery and reconfiguration in distributed systems, and distributed simulation of interconnection networks.

VANCE TYREE, *Senior Lecturer*



Mr. Vance Tyree was born in Salt Lake City, Utah and received the MSEE degree from the University of California, Berkeley in 1966. Presently he is a member of the Research Staff at USC-Information Sciences Institute. His research interests lie in large scale integrated (LSI) technology and its application to distributed computing systems, fault tolerant computing, and signal processing. He is currently studying physical processes in silicon devices that limit useful service lifetime of fine line (1.0um and smaller feature size) integrated circuits. Initially, this involves characterization of wear-out phenomena to determine a hierarchy of failure mechanisms that can be dealt with from circuit level to system architecture level. This work is ultimately aimed at providing a solid physical reliability framework upon which to build VLSI system architectures with substantially higher reliability than is possible today.

JACQUES J. VIDAL, *Professor*

Dr. Vidal was born in Liege, Belgium and received his degrees in Electrical Engineering from the University of Liege in 1951 and 1954; a post-graduate degree in Nuclear Engineering from Saclay, France in 1959 and the Doctorate from the University of Paris (Sorbonne) in 1961. He joined the UCLA faculty in 1963. In 1970 he was awarded a N.I.H. Special Fellowship to research neural coding at the UCLA Brain Research Institute and subsequently became a member of the BRI. He was originator and director of the DARPA funded Brain Computer Interface project which explored the real-time recognition of brainwave patterns. His course "Information Processing in the Nervous System" was created in 1972 and, he subsequently introduced a follow-up course on artificial neural nets and several courses on computer graphics. Profesor Vidal now heads the Distributed Machine Intelligence Laboratory. The laboratory's research focuses on massively parallel architecture for logic inferencing as well as on

several other applications of artificial neural nets to real-time dynamic control, malfunction management and sensor-event-detection.

CHAND R. VISWANATHAN, ***Professor, Electrical Engineering Department***

Dr. Chand R. Viswanathan was born in Madras, India and was educated both in India and in the United States. He received his B.S. and M.S. degrees from the University of Madras and his M.S. and Ph.D. degrees from UCLA. His teaching and research efforts are concentrated in the area of solid state and semiconductor electronics, integrated circuits and computer hardware. Dr. Viswanathan was appointed Chairperson of the Electrical Engineering Department in 1979 and completed his tour of duty June 30, 1985.

The Faculty: Publications (Calendar Year 1987)

Aoki, Masanao, *State Space Modeling of Time Series*, Springer-Verlag, Heidelberg & New York, 1987.

Aoki, Masanao, "Studies of Economic Interdependance by State Space Modeling of Time Series: US-Japan Example", *Annales a' Economie et de Statistique*, No. 5, 1987.

Aoki, Masanao, "An Alternative Measure of Random Walk Components in Time Series" *Economics Letters* vol. 24, pp. 227-230, 1987.

Aoki, Masanao, "A Convenient Framework to Analyze a Two-country World Model: An Illustrative Analysis of Anticipated Real Supply Shocks on the Exchange Rate and the Interest Rate Difference" pp.679-702 in G.Dandolfo and F.Marzano, eds., *Keynesian Theory Planning Models and Quantitiative Economics*, Guiffre, Milano, Italy, 1987.

Aoki, Masanao, "Studies of Economic Interdependence by State Space Modeling of Time Series: US-Japan Example" *Annales d'Economie et de Statistique*, N06/7 pp.225-252, 1987

Avizienis, A., "A Design Paradigm for Fault-Tolerant Systems," *Proc. AIAA Computers in Aerospace VI Conf.*, Wakefield, MA., October 1987.

Avižienis, A., "The Dependability Problem: Introduction and Verification of Fault Tolerance for a Very Complex System," *Proc. Fall Joint Computer Conference*, Dallas, TX., October 1987, pp. 89-93.

Avižienis, A., Lyu, M. R. T., Schutz, W., Tso, K.-S., Voges, U., "DEDIX 87 - A Supervisory System for Design Diversity Experiments at UCLA," in *Software Diversity in Computerized Control Systems*, U. Voges, editor, Springer-Verlag, Wien, New York, 1987, pp. 127-168.

Avižienis, A., Ball, D. E., "On the Achievement of a Highly Dependable and Fault-Tolerant Air Traffic Control System," *Computer*, Vol. 20, No. 2, February 1987, pp. 84-90.

Tso, K. S., **Avižienis, A.,** "Community Error Recovery in N-Version Software: A Design Study with Experimentation," *Digest of FTCS-17, the 17th International Symposium on Fault-Tolerant Computing*, Pittsburgh, Pennsylvania, July 1987, pp. 127-133.

Boehm, Barry, "Improving Software Productivity", *Computer*, September 1987.

Boehm, Barry, "Software Risk Management", *Proceedings, NSIA Software Risk Management Conference*, September 1987.

Boehm, Barry, "Software Environment Experience", *Proceedings, NATO Conference on Distributed Software Environments*, September 1987.

Chu, W.W. and L.M.-T. Lan, "Task Allocation and Precedence Relations for Distributed Real-Time Systems," *IEEE Transactions on Computers*, June 1987, pp. 667-679.

Chu, W.W. and K.K. Leung, "Module Replication and Assignment for Real-Time Distributed Processing Systems," *Special Issue of the IEEE Proceedings*, May 1987, pp. 547-562.

Chu, W.W. and C.M. Sit, "A Batch Service Scheduling Algorithim with Time-Out for Real-Time Distributed Processing Systems," *7th International Conference on Distributed Computing Systems*, September 1987.

Chu, W.W., K.H: Kim, and W.C. McDonald, "Testbed-Based Validation of Design Techniques for Reliable Distributed Real-Time Systems," *Special Issue of the IEEE Proceedings*, May 1987, pp. 649-666.

Chu, W.W., Guest Editor, *Special Issue on Distributed Database Systems, IEEE Proceedings*, May 1987, pp. 531-729.

J. J. Fagarasan and **J. J. DiStefano, III**, "On the Visibility of Leading Eigenvalues in Compartmental Models," *Mathematical Biosciences*, 86:97, 1987.

J. J. DiStefano, III and Viveca Sapin, "Fecal and Urinary Excretion of Six Iodothyronines in the Rat," *Endocrinology*, 121:1742, 1987.

J. J. DiStefano, III, "Optimizing Sampling Schedule Designs for Kinetic Experiments: Conceptual Framework, Software & Practical Experiences", *The Pharmacokinetics and Pharmacodynamics*, A Meeting of the USC Biomedical Simulation Resource, May 1987.

J. J. DiStefano, III, W. L. Morris, R. Wang and T. Nguyen, "Enterohepatic Regulation & Metabolism of T_3 in Hypothyroid Rats," *The American Thyroid Assn. Mtg.*, September 1987.

Dyer, M.G., Cullingford, R. & Alvarado, S., "Scripts", *Encyclopedia of Artificial Intelligence*, Shapiro (ed.), John Wiley & Sons, 1987.

Dyer, M.G., "Emotions and Their Computations: Three Computer Models", *Cognition and Emotion*, Vol 1, No. 3, pp. 323-347, 1987.

Dolan, C. and **M.G. Dyer**, "Towards the Evolution of Symbols", *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications*, Cambridge, MA, July 1987.

Hodges, J., **Flowers, M. and M.G. Dyer**, "Knowledge Representation for Design Creativity", *Proceedings of Winter Annual Meeting of American Society of Mechanical Engineers (ASME-87)*, Boston, MA, December 1987.

Dolan, C. and **M.G. Dyer**, "Symbolic Schemata, Role Binding and the Evolution of Structure in Connectionist Memories", *Proceedings of the 10th Intl. Joint Conference on Artificial Intelligence (IJCAI-87)*, Milan, Italy, August 1987.

Dolan, C. and **M.G. Dyer**, "Evolution of an Architecture for Symbol Processing", and the Evolution of Structure in Connectionist Memories", *Proceedings of IEE First Annual International Conference on Neural Networks*, San Diego, CA, June 1987.

Pazzani, M. and **M.G. Dyer**, "A Comparison of Concept Identification in Human Learning and Network Learning with the Generalized Delta Rule", *Proceedings of the 10th Intl. Joint Conference on Artificial Intelligence (IJCAI-87)*, Milan, Italy, August 1987.

Pazzani, M. **M.G. Dyer, and M. Flowers**, "Using Prior Learning to Facilitate the Learning of New Causal Theories", *Proceedings of the 10th Intl. Joint Conference on Artificial Intelligence (IJCAI-87)*, Milan, Italy, August 1987.

Goldman, S.R., **Dyer, M.G. and M. Flowers**, "Precedent-based Legal Reasoning and Knowledge Acquisition in Contract Law", *Proceedings of the First Intl. Conference on Artificial Intelligence and Law (ICAIL-87)*, Boston, MA, May 1987. (Sponsored by ACM SIGart and Northeastern Univ. Center for Law and Computer Science).

Dyer, M.G., Flowers, M. and J. Hodges, "Naive Mechanics Comprehension and Invention in EDISON", *Proceedings of the 10th Intl. Joint Conference on Artificial Intelligence (IJCAI-87)*, Milan, Italy, August 1987.

Alkalaj,L., **M.D. Ercegovac**, and T. Lang, "A Dynamic Memory Management Policy for FP", 1987 Hawaii International Conference on Systems Science.

Ercegovac, M.D. and T. Lang, "On-line Scheme for Computing Rotation Factors", Proc. 8th IEEE Symposium on Computer Arithmetic, 1987.

Tu, P. and **M.D. Ercegovac**, "A Radix-4 On-Line Division Algorithm", 8th IEEE Symposium on Computer Arithmetic, 1987.

Ercegovac, M.D. and T. Lang, "On-the-Fly Conversion of Redundant into Conventional Representations", IEEE Transactions on Computers, Vol. C-36, No.7, July 1987, pp.895-897.

Ercegovac, M.D. and T. Lang, "On-Line Schemes for Computing Rotation Angles for SVDs", Proc. SPIE Conference on Real-Time Signal Processing, San Diego, August 1987.

Ercegovac, M.D., T. Lang, J.G. Nash, "An Area-Time Efficient Binary Divider", Proc. ICCD '87 Conference, New York, 1987.

Ercegovac, M.D. and T. Lang, "Fast Radix-4 Multiplication Without Carry-Propagate Addition", Proc. ICCD '87 Conference, New York, 1987.

Kapelnikov, A., R.R. Muntz, and **M.D. Ercegovac**, "A Methodology for the Performance Evaluation of Distributed Computations", Proc. IFIP Conference on Distributed Processing, October 1987.

Ravi, T.M., **M.D. Ercegovac**, T. Lang and R.R. Muntz, "Static Allocation for a Dataflow Multiprocessor System", Proc. 2nd International Conference on Supercomputing, Santa Clara, 1987, pp.169-178.

Ercegovac, M.D. and T. Lang, "Fast Cosine/Sine Algorithm Using On-Line Cordic", IEEE Asilomar Conference on Signals, Systems, and Computers, 1987.

Nash, J.G., L.W. Chow, **M.D. Ercegovac**, and T. Lang, "Implementation of a Serial/Parallel Multiplier and Divider on a Systolic Chip", IEEE Asilomar Conference on Signals, Systems, and Computers, 1987. Publications

Estrin, Thelma, "The UCLA Brain Research Institute Data Processing Laboratory", *Conference Proceedings*, The ACM Conference on the History of Medical Informatics, Bethesda, Maryland, November 5-6, 1987.

Gerla, M. and G.S. Wang, "Performance Models of Buzz-Net, A Hybrid Fiber Optics LAN", *INFOCOM '87 Proceedings*, San Francisco, CA, April 1987.

Gerla, M., G.S. Wang, and P. Rodrigues, "Buzz-Net: A Hybrid Token/Random Access LAN", *IEEE JSAC*, July 1987.

Gerla, M. and C. Yeh, "Interconnection of Fiber Optics LAN's" *SPIE Symposium on Fiber Optics*, August 1987.

Gerla, M., "Routing and Flow Control in Processor Limited Packet Networks", *URSI Conference*, Tel Aviv, August 1987.

Inselberg, A., and B. Dimsdale, "Parallel Coordinates for Multi-Dimensional Grapics", *Proceedings, National Computer Graphics Assoc, National Conference*, Philadelphia, PA, (March 1987), pp. 547-566. (Invited paper - received Best Paper Award).

Inselberg, A., B. Dimsdale, "Parallel Coordinates for Visualizing Multi-Dimensional Geometry", *Proceedings of 5th International Computer Graphics Conference*, T. Kunii (ed.), Springer-Verlag, Tokyo, 1987, pp. 25-44.

W. J. Karplus, (Editor), "Multiprocessors and Array Processors", Society for Computer Simulation, San Diego, California, Simulation Series, Vol. 18, No. 2, 1987.

W. J. Karplus "Parallelism and Pipelining: The Road to More Cost-Effective Scientific Computing", *Simulation Series* Society for Computer Simulation, Vol. 18, No. 2, pp 1-13, January 1987.

R. Tomovic, G.A. Bekey and **W. J. Karplus**, "A Strategy for Grasp Synthesis with Multifingered Robot Hands", *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 83-89, Raleigh, North Carolina, March 1987.

A. Makoui and **W. J. Karplus**, "ALI: A CSSL/Multiprocessor Software Interface", *Simulation*, Journal of the Society for Computer Simulation, Vol 49, No. 2, pp 63-71, August 1987.

W. J. Karplus, "Can Simulation Enhance the Credibility of Plans for National Defense?", Summer Simulation Conference, Montreal, Canada, July 1987.

Kleinrock, L. and J. Silvester, "Spatial Reuse in Multihop Packet Radio Networks," invited paper for *Proceedings of the IEEE*, Special Issue on Packet Radio Networks, Vol. 75, No. 1, January 1987, pp. 156-167.

Takagi, H. and L. **Kleinrock**, "Correction to 'Throughput Analysis for Persistent CSMA Systems'," Correspondence Item, *IEEE Transactions on Communications*, Vol. COM-35, No. 2, February 1987, pp. 243-245.

Kleinrock, L. and H. Levy, "On the Behavior of a Very Fast Bidirectional Bus Network," Proceedings of the *IEEE International Conference on Communications (ICC '87)*, June 1987, pp. 1419-1426.

Lewis, C., Bourbakis, N. and **Klinger, A.**, "An Efficient Implementation Scheme for the SCAN Language," presented August 1987, IEEE Workshop on Languages for Automation, Vienna, Austria.

Paul, J., Kilgore, T.E., and **Klinger, A.**, "New Algorithms for Automated Symmetry Recognition", Advances in Intelligent Robotics Systems -- SPIE's Cambridge Symposium on Optical and Optoelectronic Engineering, 1-6 November 1987, Cambridge, Massachusetts.

Klinger, A., "Data Structures", *The Encyclopedia of Physical Science and Technology, Vol. 4*, Meyers, R. A. (ed.), Academic Press, Inc., New York, 1987, pp. 125 - 135.

Baraghimian, G. and **Klinger, A.**, "Hexagonal Decomposition: A Data Structure for Planar Image Tesselation," *Proceedings 1987 IEEE International Conference on Systems, Man, and* Cybernetics, Piscataway, New Jersey 08854, IEEE Service Center, pp. 1011 - 1014, 1987.

Korf, Richard, "Search," in the *Encyclopedia of Artificial Intelligence*, John Wiley, New York, 1987, pp. 994-998.

Korf, Richard, "Heuristics," in the *Encyclopedia of Artificial Intelligence*, John Wiley, New York, 1987, pp. 376-380.

Korf, Richard, "Planning As Search: A Quantitative Approach," *Artificial Intelligence*, Vol. 33, No. 1, pp. 65-88, 1987.

Korf, Richard and J. Pearl "Search Techniques," in *Annual Review of Computer Science*, Vol. 2, Annual Reviews Inc., Palo Alto, Ca., 1987.

Korf, Richard and B. Abramson "A Model of Two-player Evaluation Functions, *Proceedings of the National Conference on Artificial Intelligence (AAAI-87)*, Seattle, Wash., July, 1987, pp 90-94.

Korf, Richard "Real-time Heuristic Search: First results," *Proceedings of the National Conference on Artificial Intelligence (AAAI-87)*, Seattle, Wash., July, 1987, pp. 133-138.

Korf, Richard, "Real-time Path Planning," in *Proceedings of the DARPA Workshop on Knowledge-Based Planning*, Austin, Tx., Dec. 1987.

M. Tremblay and **T. Lang**, " Implementation of a Shift-Register Register File", Proceedings of the 20th Annual Hawaii International Conference, January 1987

L. Alkalaj, M. Ercegovac, and **T. Lang**, "A Dynamic Memory Management Policy for FP", Proceedings of the 20th Annual Hawaii International Conference, January 1987.

T. Ravi, M. Ercegovac, **T. Lang**, and R. Muntz, "Static allocation for a Data Flow Multiprocessor System", *Proceedings of the 2nd International Conference on Supercomputers*, San Francisco, March 1987.

M. Ercegovac and **T. Lang**, "On-Line Scheme for Computing Rotation Factors", Proceedings 8th Symposium on Computer Arithmetic, pp. 196-203, May 1987.

M. Huguet, **T. Lang**, and Y. Tamir, "A Block-and-Actions Generator as an Alternative to a Simulator to Collect Architecture Measurements," Proceedings SIGPLAN'87 Symposium on Interpreters and Interpretive Techniques, June 1987.

M. Ercegovac and **T. Lang**, "On-the-Fly Conversion of Redundant into Conventional Representations", IEEE Transactions on Computers, Vol. C-36, No.7, July 1987, pp. 885-887.

J. Moreno and T. Lang, " Design of Special-Purpose Arrays for Matrix Computations: Preliminary Results", Proceedings of SPIE, Vol. 827, pp. 53-65, August 1987.

M. Ercegovac and **T. Lang**, "On-Line Schemes for Computing Rotation Angles for SVDs", Proc. of SPIE, Vol. 826, August 1987.

M. Ercegovac, **T. Lang**, G. Nash, and L. Chow, "An Area-Time Efficient Binary Divider", Proceedings IEEE International Conference on Computer Design, pp. 645-648,October 1987.

M. Ercegovac and **T. Lang**, "Radix-4 Multiplication Without Carry-Propagate Addition", Proceedings IEEE International Conference on Computer Design, pp.654-658, October 1987.

M. Ercegovac and **T. Lang**, "Fast Cosine/Sine Implementation using On-Line CORDIC", Proceedings 21st Annual Asilomar Conference on Signals, Systems, and Computers, November 1987.

Kinsey, R.J., C.H. McKenzie and **P.H. Mak**, "Autonomous Gyro Bias Detection and Identification Algorithm for the DMSP Spacecraft," *AIAA Guidance Navigation & Control Conference*, Monterey, CA, August 17-19, 1987.

Tong, M.M, C.H. Smith and **P.H. Mak**, "On-Orbit Balancing of a Spinning Antenna", *AAS/AIAA Astrodynamics Specialist Conference*, Kalispell, Montana, August 10-13, 1987.

Mak, P.H, M.M. Tong and A.B. Jenkins, "Dynamics and Control Analysis of a Satellite with a Large Flexible Spinning Antenna," *AAS/AIAA Astrodynamics Specialist Conference*, Kalispell, Montana, August 10-13, 1987.

Melkanoff, M.A., "Computer-Aided Design," *Encyclopedia of Artificial Intelligence, Volume 1.*, Eds., S. C. Shapiro and D. Eckroth, John Wiley & Sons, Wiley-Interscience Publications, pp. 151-153, 1987.

Melkanoff, M.A., Kops, L., Lichten, L. "Application of CAD Modeling for Information Enhancement in Experimental Flow Pattern Studies", *Proc. Manufacturing Systems Fertigungssysteme Systemes de Fabrication*, CIRP Annals, Vol 36/1, 1987.

Bond, A., **Melkanoff, M.A.**, et al, "Automatic Extraction of Geometric Features from CAD Models," *Proc. of 2nd International Conf. on Intelligent Manufacturing Systems*, Dubrovnik, Yugoslavia, August 24-29, 1987 (invited)

Roberti, P., **Melkanoff, M.A.** "Seff-Adaptive Stress Analysis Based on Stress Convergence," *International Journal for Numerical Methods in Engineering*, John Wiley & Sons, Wiley-Interscience Series, 1987.

Melkanoff, M.A. "Education for Intelligent Manufacturing Systems," *Robotics - Integrated Manufacturing*, Vol. 3, No. 2., pp. 165-169, 1987, 0736-5845/87, Pergamon Journals, Ltd. (GB).

Ravi,T.M., Ercegovac,M.D., Lang,T., **Muntz,R.R.**, "Static Allocation for a Data Flow Multiprocessor System," *Proceedings of the 2nd Int. Conf. on Supercomputers*, San Francisco, CA, pp.1-25, 1987

de Souza e Siva,E., **Muntz,R.R.** "Approximate Solutions for a Class of Non-Product Form Queueing Networks", *Performance Evaluation*, August,1987, pp. 221-242.

Samadi,B., **Muntz,R.R.**, Parker,D.S., "A Distributed Algorithm to Detect a Global State of a Distributed Simulation System", *IFIP Conference on Distributed Processing*, Amsterdam, Oct. 1987.

Kapelnikov,A., **Muntz,R.R.**, Ercegovac.M.D., "A Methodology for the Performance Evaluation of Distributed Computations", *IFIP Conference on Distributed Processing*, Amsterdam, Oct. 1987.

Geffner, H. and **Pearl, J.**, "A Distributed Diagnosis of Systems with Multiple Faults" *Proceedings, 3rd IEEE Conference on Artificial Intelligence* Applications, Orlando, Florida, February 1987, pp. 156-162.

Dechter, R. and **Pearl, J.**, "The Cycle-Cutset Method for Improving Search Performance in AI Applications," *Proceedings, 3rd IEEE Conference on Artificial Intelligence* Applications, Orlando, Florida, February 1987, pp. 224-230.

Pearl, J., "Evidential Reasoning Using Stochastic Simulation of Causal Models," *Artificial Intelligence*, Vol. 32:2, 1987, pp. 245-258.

Pearl, J., "Bayes Decision Methods," "Branching Factor," "Game Trees" & "AND/OR Graphs," *Encyclopedia of AI*, Wiley Interscience, New York, 1987.

Kim, J. and **Pearl, J.**, "CONVINCE: a CONVersational INference Consolidating Engine," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-17(2), 1987, pp. 120-132.

Pearl, J. "Embracing Causality in Formal Reasoning." *Proceedings*, AAAI Conference, Seattle, Wash. July 1987, 369-373.

Pearl, J., and Verma,T., "The Logic of Representing Dependencies by Directed Graphs," *Proceedings*, AAAI Conference, Seattle, WA. July, 1987, pp. 374-379.

Rebane, G., and **Pearl, J.**, "The Recovery of Causal Poly-Trees from Statistical Data," *Proceedings*, AAAI Workshop on Uncertainty in AI, Seattle, WA. July 1987, pp. 222-228.

Xu, Lei and **Pearl, J.**, "Structuring Causal Tree Models with Continuous Variables," *Proceedings*, AAAI Workshop on Uncertainty in AI, Seattle, WA. July 1987, pp. 170-179.

Judea Pearl, "Do We Need Higher-Order Probabilities and, If so, What do they Mean? *Proceedings*, AAAI Workshop on Uncertainty in AI, Seattle, WA. July 1987, pp. 47-60.

Geffner, H. and **Pearl, J.**, "An Improved Constraint-Propagation Algorithm for Diagnosis." *Proceedings*, IJCAI Conference, Milano, Italy, August 1987, pp. 1105-1111.

Pearl, J., and Korf, R., "Search Techniques," *Annual Review of Computer Science*, Vol II, 1987.

Geffner, H. and **Pearl, J.**, "On the Probabilisitc Semantics of Connectionist Networks," *Proceedings*, Ist IEEE International Conference on Neural Networks, San Diego, CA. June 1987.

Dechter, R. and **Pearl, J.**, "Network-Based Heuristics for Constraint-Satisfaction Problems," *Artificial Intelligence*, Vol. 34:1, December 1987.

Rennels, D., "Design Issues in High-Performance Fault-Tolerant Multicomputers," *Proc. 3. Internationale GI/ITG/GMA- Fachtagung, (in Informatik-Fachberichte #147 Springer Verlag)* Bremerhaven, Germany, September 1987, pp. 51-60.

Rennels, D., "Fault-Tolerant Computing: Issues, Examples, and Methodology," *Lecture Notes, Advanced Workshop on Fault-Tolerant Computing*, Bangalore, India, July 20-25, 1987, pp. 1-58.

Rennels, D., "The Evolution of a Fault-Tolerant Design: Contractor Milestones and Evaluation," *Proc. AIAA Computers in Aerospace VI Conference*, Wakefield, Mass., October 1987, pp. 58-70.

Avizienis, A., and **Rennels, D.** "The Evolution of Fault Tolerant Computing at the Jet Propulsion Laboratory and at UCLA," *in Dependable Computing and Fault-Tolerant Systems Vol 1. The Evolution of Fault-Tolerant Computing*, Springer-Verlag, 1987, pp. 141-192.

Gungner,D., and **J. Skrzypek**, "UCLA PUNNS: Neural Network Machine for Computer Vision", *Proceed. of Image Understanding Conf.* IEEE, SPIE and Pattern Recog. Soc., Los Angeles, CA, 1987.

Paik,E., D. Gungner, and **J. Skrzypek**, "UCLA SFINX - A Neural Network Simulation Environment", *IEEE First Annual Conference on Neural Networks*, San Diego, CA, 1987.

Skrzypek, J., "Sensing and Perception: Connectionist Approaches to Subcognitive Computing", *NASA Proceedings, JPL Space Telerobotics Workshop*, 1987.

Heisey, L, and **Skrzypek,J.** "A Neural Netwrok Model for Early Color Vision: A Neural Network Architecture", *IEEE First Annual Conference on Neural Networks*, San Diego, CA. 1987.

Skrzypek, J., and E. Mesrobian, "Textural Segmentation: Gestalt Heuristics as a Connectionist Hierarchy of Feature Detectors", *9th IEEE Annual Conference in Med. and Bio. Soc.*, Boston, 1987.

Mesrobian, E., and **J. Skrzypek**, "Connectionist Structure for Computing Textural Segmentation", *Proceedings of the Conf. on Image Understanding*, IEEE, SPIE and Pattern Recognit. Soc., Los Angeles, CA, 1987.

Mesrobian, E., and **J. Skrzypek**, "Discrimination of Higher-order Textures: A Neural Network Architecture", *IEEE First Annual Conference on Neural Networks*, San Diego, CA. 1987.

Stiber, M. and J. Skrzypek, "LISP-based PC Vision Workstation",
Proceeding of the Conf. on Image Understanding, IEEE, SPIE, and Pattern Recognition Soc., Los Angeles, CA, 1987.

C. H. Séquin and Y. Tamir, "Fault Tolerant VLSI Multicomputers," pp. 429-449 in *VLSI CAD Tools and Applications*, ed. W. Fichtner and M. Morf, Kluwer Acadmic Publishers, (1987).

M. Huguet, T. Lang, and Y. Tamir, "A Block-and-Actions Generator as an Alternative to a Simulator for Collecting Architecture Measurements," *SIGPLAN'87 Symposium on Interpreters and Interpretive Techniques*, St. Paul, MN, pp. 14-25, (June 1987).

Y. Tamir and E. Gafni, "A Software-Based Hardware Fault Tolerance Scheme for Multicomputers," *1987 International Conference on Parallel Processing*, St. Charles, IL, pp. 117-120, (August 1987).

The Faculty: Honors and Awards in 1987

The Computer Science Department is pleased to announce that Professor Gerald Estrin has been elected to the rank of Fellow of the American Association for the Advancement of Science (AAAS). A Fellow of the AAAS is defined as "a Member whose efforts on behalf of the advancement of science or its applications are scientifically or socially distinguished." The citation reads as follows:

> "For Research leading to innovative and pioneering computer systems, and for research on use of computers in the design process itself."

In this year also, Professor **Michel Melkanoff** was honored by the San Fernando Valley Engineers' Council with the Distinguished Engineering Education Achievement Award for 1987.

Professor **Algirdas Avižienis** was the recipient of the Thomas R. Benedict Memorial Award for the outstanding paper of the AIAA Computers in Aerospace VI Conference, Wakefield, Massachussetts, October 7-9, 1987.

For his invited paper, "Parallel Coordinates for Multidimensional Graphics", co-authored with B. Dimsdale, Professor **Alfred Inselberg** received the Best Paper Award at the National Computer Graphics Association Conference.

One of our students', **Dagan Feng**, who is working under the supervision of Dr. J. DiStefano, was honored with the Crump Institute of Medical Engineering Award.

Another of our students, **Rina Dechter**, who received her Ph.D. under the supervision of Professor J. Pearl, was granted a renewal of her University of California Presidential Postdoctoral Fellowship.

COMPUTER SCIENCE DEPARTMENT
TEACHING ASSISTANTS, RESEARCH STAFF AND FELLOWSHIP HOLDERS

The Computer Science Department has more than 270 enrolled graduate students. Each year a large proportion of these students are awarded Teaching Assistantships, Research Assistantships, or Fellowships or are hired in technical staff titles. The people included below form this year's cadre of supported graduate students and technical staff.

DEC Fellow - Christina Aguilera.

Departmental Fellows - Claus Cooper, Thomas Funkhouser, Michael Rowley, Rony Sawdayi.

Hughes Fellows - Tamir Baraad, Robert C. Boehnlein, Ricardo B. Cabral, William Lincoln, Marti P. Sheldon, Kurt A. Stoll, Stephanie August, Charles P. Dolan, John Maslow, John Oyster, Gregory Baraghimian, Laurie O'Connor, Inge Heisey, Kirk D. Koyamatsu, Frank D. Schillinger.

IBM Fellows - Arthur Goldberg, Jaime Moreno, Johanna Moore, Thirumalai Ravi.

MICRO Fellows - Michael Coleman, Marco Ma, Jason Rosenberg, Charles Tong.

Teaching Assitants - Hsin-Chou Chi, Robert J. Collins, David B. Dobrikin, Sanjay Jain, Edmund K. Kwan, Edwin R. Tisdale, David S. Wine.

Teaching Associates - Stephen G. Faris, Jeong-A Lee, Ting Y. Leung, Chi Shing Lui, Jose J. Miro-Julia, Grabriel Robins, Rony Ross, John Tobias, Charles Hokshun Tong, Tella N. Vijayakumar, Gregory L. Wong.

Research Assistants - Leonid V. Belyaev, Jia-Hong Chen, Jae Chun Cho, Gregory L. Frazier, Tiffany M. Gaber, Ming-Yun Horng, Milan M. Kovacevic, Kong Li, Tzung-I Lin, Quyen D. Ngo, Ramachandra Prasadram, Michael D. Stiber, Peter G. Trajmar, Ping-Hann Wang.

Post Graduate Research Engineers - Leon Alkalaj, Alberto Avritzer, Steven E. Berson, Jeffrey Chang, Keh-Jeng Chang, Hau-Ming L. Chau, Li-Wen Chen, William C. Cheng, Ching-Tsun Chou, John S. Eickmeyer, Robert E. Felderman, Christopher P. Ferguson, Brett D. Fleisch, Michael E. Gasser, Hector A. Geffer, Dan Geiger, Arthur P. Goldberg, Moises Goldzmidt, Joseph Green, Daniel Greening, David Gunger, John Hodges, Ming-Yun Horng, Jau-Hsiung Huang, Miquel Huguet, Richard Huntsinger,Scott Kalter, Hyeongil Kim, Ted Kim, Willard Korfhage, Chi Young Ku, Weng-Ling Kuo, Lance Kurisaki, Dorothy Landis, Koenraad Lecot, Geunbae Lee, Rei-Chi Lee, Poman Leung, Ting Leung, Kong Li, Randall Lichota, Ximming Lin, Brian Livezey, Wen-Jeng Lue, Rung-Tsong Lyu, Farid Mehovic, Itay Meiri, Mesrobian Edmond, Risto Miikkulainen, Jose Monteiro, Jaime Moreno, Valery Nenov, Hing-Sun Ngai, Thomas Page, Joseph Pemberton, Curt Powley, Maria Pozzo, Thirumalai Ravi, George Rebane, Bruce Rosen, Rony Ross, Eve Schooler, Chung-Dak Shum, Chi-Man Sit, Scott Spetka, Ronald Sumida, Yea-Li Sun, Li-Ken Tang, Marc Tremblay, Kang-Guo Tu, Thomas Verma, George Wang, Yih-Jih Wang, Chi-Sharn Wu, Winthrop Wu, Hsing-Chien Yeh.

Following are biographies of some of our graduate students.

Christine S. Ruiz Aguilera was born and raised in Pasadena, California. She received her B.A. in Mathematics from U.C. Irvine (1970), her M.A. in Mathematics from U.C. San Diego (1972), and her M.S. in Computer Science from UCLA (Fall 1987) under the supervision of Prof. Dan Berry in the area of Software Systems. The subject of her thesis was: "Finding Abstractions in Problem Descriptions Using findphrases". Since 1972 she has been a full-time instructor of Mathematics for Moorpark College, a community college in Ventura County where from 1983 to 1985 served as Chairperson for the Mathematics/Computer Science Department. Christine is currently on-leave from Moorpark in order to pursue her doctorate in the field of Formal Semantics of Programming Languages. She is a Digital Equipment Corporation Fellowship recipient (1986-88) and is presently serving as the C.S. Department's Undergraduate Student

Affairs Officer. Christine is a member of the Mathematical Association of America and the Association of Women in Mathematics. Her other interests include racquetball and skiing.

Leon Alkalaj was born in Belgrade, Yugoslavia, on Christmas Day 1958. He received his Electrical Engineering Diploma in 1982 from the Faculty of Electrical Engineering, University of Belgrade. He has completed his Master's degree and is pursuing a Ph.D. degree. His main interests are in the field of computer architecture with a special emphasis on non-Von Neumann machines, functional language implementation and VLSI design. In his spare time he plays tennis, soccer and supports the Red Star Belgrade Soccer team.

Colin Allen is pursuing a Ph.D. in philosophy and a M.S. in computer science. He has T.A.-ed several philosophy courses as well as an Honors division introduction to Lisp and Artificial Intelligence. Hobbies are bicycling (road and off-road) and scuba diving. Research interests are artificial intelligence and animal cognition.

Sergio J. Alvarado is a Ph.D. candidate in the Computer Science Department at UCLA, where he is pursuing research on artificial intelligence. He received the Electronics Engineer degree from Simon Bolivar University, Venezuela, in 1978; the M.Sc. degree in Electrical Engineering (digital systems) from Central University of Venezuela in 1981; and the Engineer degree from UCLA in 1984 in the field of artificial intelligence with minor fields in data base management and biocybernetics. From 1982 to 1984 he taught Engineering 11 (Introduction to Patterns of Problem Solving) at the UCLA School of Engineering and Applied Science. Since then he has been a postgraduate research engineer at the UCLA Artificial Intelligence Laboratory under Professor Michael G. Dyer. His Ph.D. dissertation focuses on the knowledge sources and processes required for computer comprehension of opinions, arguments, and issues which arise in politico-economic editorials. His research interests include: natural language processing, human reasoning and argumentation, belief systems, knowledge representation, and machine learning. Mr. Alvarado was the recipient of the 1985 International Joint Conference on Artificial Intelligence (IJCAI) Doctoral Fellowship. He is a member of IEEE (Computer Society), ACM (SIGART), AAAI, ACL, and the Cognitive Science Society.

Stephanie August is working on the PhD degree in Computer Science with a specialization in Natural Language Processing. Her research interests include analogical reasoning, natural language processing, argumentation, cognitive modelling, and intelligent data management systems. Stephanie's research focuses on developing a computer model of the process of understanding analogies in the domain of editorial letters. Her minor fields are Software Systems/Data Bases and Computer Science Theory. Stephanie works for Hughes Aircraft Company, where she is a Hughes Doctoral Fellow. At Hughes Stephanie is currently supporting an image understanding project. At Hughes she has also done research in natural language processing, and developed a prototype hybrid data base management system combining a frame-based inference engine with a relational database manager. Stephanie earned her MS in Computer Science with a specialization in Artificial Intelligence from UCLA in 1985. Her thesis was entitled "Analogy Recognition and Comprehension in Editorials". She earned her BA in Slavic Languages from UCLA in 1972. Stephanie is a member of the IEEE Computer Society, ACM, ACL, AAAI, the Association for Automated Reasoning, and the Cognitive Science Society. She also enjoys raising her three daughters, renovating the house with her husband, assisting with her daughter's Brownie Girl Scout troop, gardening, camping, and playing folk guitar.

Alberto Avritzer was born in Belo Horizonte, State of Minas Gerais, Brazil. He received the B.Sc in Computer Engineering in 1980 from the Technion and the M.S. in Computer Science in 1983 from the Federal University of Minas Gerais(UFMG). From 1981 to 1986 he worked as a Research Assistant at the Computer Science Department of UFMG . He came to UCLA in the fall of 1986 to pursue his Ph.D. In the summer of 87 he went IBM T.J. Watson Research Center where he did research on modeling and simulation of Distributed Systems. He is currently employed as a Research Assistant in the IBM joint studies project. His main interests are Modeling of Distributed Systems , Data Bases and Computer Architecture. Among his extra-curricular interests are photography, sailing, bridge and soccer playing.

Tamir A. Baraad was born in Caracas, Venezuela and raised in Tel Aviv, Israel. He obtained his B.S. degree in Computer Science from New York University in 1985 and is now studying towards an M.S. degree. His main interests are in the area of programming languages and systems. He also works for Hughes Aircraft Company and is a Hughes fellowship recipient. His other interests include film and Israeli folk dancing.

Peter Berke received his B.S. in Electrical Engineering and Computer Science from M.I.T. in 1978 and received his M.B.A. from UCLA in 1985. His interests include: array theory, art, communication, complexity, computation, concepts, conflict, creativity, dance, dancing, decisions, design, graphs, juggling, knowledge, language, life, logic, management, math, mime, motion, movement, networks, statistics, surveys, and systems (social, formal, and computer; biological, conceptual, electrical, living, physical, and formal). His motto is "Bring back disco."

Steve Berson received his B.S. degree in Computer Science from the University of Wisconsin in 1981. His research interests are designing advanced mathematical modeling environments, especially for, but not restricted to computer and communication systems. Other interests include hiking and badminton. Steve is also a co-founder and former vice president of The Engineers for Social Interaction (ESI).

Joseph Betser was born in Tel Aviv, Israel, and grew up in Haifa for the most part. He completed his undergraduate studies at the TECHNION - Israel Institute of Technology, graduating with honors in 1976, and receiving his B.Sc. from the Department of Aeronautical Engineering. Starting in 1977 he served in the Israeli Air Force, completing his term in 1981 at the rank of Captain. Mr. Betser joined UCLA's M.S. program in the Computer Science Department in September 1981 as a School of Engineering Fellowship recipient, and was subsequently selected for the 1982/83 Hortense Fishbaugh campus wide scholarship. During that period, Joe taught Operating System courses at UCLA. Joe completed his M.S. in 1984, and is currently a Ph.D. candidate. He has research interests in large and heterogeneous distributed systems, performance analysis, simulation and modeling, and engineering related aspects of computer science. Joseph Betser is pursuing his research interests under the supervision of Dr. Karplus and Dr. Carlyle. One object of study under the IBM-UCLA Joint Study contract is the School of Engineering and Applied Sciences Network (SEASnet). This heterogeneous distributed system provides us with exciting analysis, measurement, and synthesis prediction opportunities. Leisure time interests include photography, travel, people, family affairs, and other sources of thrilling experience.

Steve Breese was born in Tucson, Arizona. He received his B.S. in Engineering here at U.C.L.A. in June 1987, and is continuing toward an M.S. degree in Scientific Computing with emphasis on applications to modeling biological systems. Steve is concurrently working at the Aerospace Corporation, where he has been granted a fellowship to continue his studies. At Aerospace, he is occupied primarily with estimation and real-time simulation of satellite systems.

Gil R. Case was born in Littlefield, Texas in 1943. He completed four years of undergraduate work in math and physics at CSULB. He worked in industry for 17 years for such companies as GE, Hughes, CalComp, CSC, and TRW and has had three of his own businesses. He received his MBA from National University, San Diego in 1985. Gil has taught speed reading and development, has taught 16 computer courses, assisted and coordinated 15 professional seminars, and has written several internal commercial books. Gil's research interests include the understanding of human learning and its potential and artificial neural systems. Among his extra-curricular interests are photography, skiing, scuba diving, windsurfing, and jogging.

Keh-jeng Chang was born in Jia(-) Yi(), Southern Taiwan. He received his B.S. and M.S. from the Department of Electrical Engineering, National Taiwan University. He came to UCLA CSD in the Fall of 1983. Currently, he is a Ph.D. student working for Prof. McNamee on the design and implementation of a CAD system to generate schematic diagrams for VLSI circuits. His areas of interest include VLSI design automation, AI, distributed systems and computer networks. His outside interests are table tennis, badminton, swimming and debating with others about politics.

Meng Chang Chen was born in Taipei, Taiwan. He received B.S. and M.S. degree in Computer Science from National Chiao-Tung University, Hsinchu, Taiwan. He joined UCLA as a Ph.D.

student in October 1983. He served as TA of CS141 and CS181 courses, and RA at times. Currently, he works for Dr. McNamee in Statistical Databases. His interests and experiences include knowledge bases, statistical databases, logic programming, knowledge representation, inexact reasoning, and database management.

Fei-Ming Chen was born and raised in Taiwan. She moved to Japan in 1981 and completed high school education in Tokyo two years later. In 1983, she came to study at UCLA as a Math/Computer Science student and received her B.S. degree in 1987. She is now working on her M.S. in Computer Science. She is currently working at the AIDS Clinical and Research Laboratory of UCLA Medical Center as a programmer.

William (Bill) Cheng was born in Taiwan. He received his B.S. degree in Electrical Engineering from UCLA in July 1981. He then got his M.S. degree in Electrical Engineering from Stanford University in July 1982. After a year of working in the industry, Bill decided to come back to school in 1983 to work on his Ph.D. His academic interests include: modeling of computer systems, logic programming, computer graphics, and computer architecture.

Hsin-Chou Chi was born in Taiwan, Republic of China, in 1960. He received his B.S. and M.S. degrees from the Department of Electrical Engineering, National Taiwan University in 1982 and 1984 repectively. After two years of serving in the army, he came to UCLA Computer Science Department. His interests are parallel and distributed processing, and fault tolerance.

Ching-Tsun Chou was born on August 11, 1961 in Taipei, Taiwan. He received his B.S. degree in Electrical Engineering from National Taiwan University in June, 1983. Then he was admitted to the graduate program of Computer Science in UCLA in the same summer. He finished his M.S. degree at the end of 1985. Pursuing his Ph.D. degree with an emphasis in theoretical computer science, he is now a research assistant with Professor Eli Gafni.

Michael Coleman was born in Kansas City, Missouri in 1964. He studied Computer Science at the University of Missouri at Kansas City where he worked on a diagnostic expert system as a research assistant for Professor Steven Graham. He graduated with B.S. degrees in Computer Science and Mathematics in December, 1986 and entered UCLA that fall supported by a MICRO Fellowship. His current research interests are in the areas of artificial intelligence and software engineering. In his spare time, he reads, follows world events, and marvels at the awesome chutzpah of Los Angeles drivers.

Rob Collins was born in 1965 in Manchester, CT, although he spent most of his formative years living in idyllic Apple Valley, MN. He received his A.B. (cum laude) from Dartmouth College in 1987, with High Honors in Computer Science. His academic interests include computational geometry (especially Voronoi diagrams), programming languages, operating systems, and parallel and distributed systems. His other interests include fencing (sabre in particular), hiking, skiing, frisbee, travel, and anything else that can be done outdoors.

Claus Cooper is a new Ph.D. student interested in several areas of Computer Science, including distributed systems and algorithms. He was born in Champaign, Illinois, but was abruptly transferred to a different part of the country, eventually growing up in Colorado Springs, Colorado. Claus attended Carnegie Mellon University, and in 1987 was awarded a B.S. in Applied Mathematics. He is now at UCLA courtesy of a MICRO Fellowship.

Maneesh Dhir was born in New Delhi, India in 1964. He received his Bachelor of Technology degree in Electrical Engineering in June 1986. He has been working with the SARA/IDEAS group since Sept. '86, and is currently employed as a Post-Graduate Research Engineer. As part of his research work he is currently working on the development of a Performance Analysis tool for the SARA system. His areas of interest include performance analysis, simulation & modeling of computer Architectures, and Networks.

David Dobrikin was born in Inglewood, CA. He has lived in Vancouver, Canada for the last sixteen years. He studied Computer Science at the University of Washington and graduated with a B.S. with College Honors in June 1987. He was also accepted into the Phi Beta Kappa honor society. David joined UCLA in the Fall of 1987. He is currently pursuing a Master's degree in

the architecture field and is a Teaching Assistant for CS151A. His academic interests include: computer architecture, VLSI design and computer science theory.

John Eickemeyer was born in Glendale, CA in 1961, and is a native of Los Angeles. He received his B.A. in Mathematics from Occidental College in 1983, and completed his M.S. degree in Computer Science (specializing in Computer Graphics) at U.C.L.A. in 1987. John's interests are numerous and diverse, and include mathematics, computer science, music (both choral and instrumental), drama and theatre, games (mathematical, video, fantasy, etc.), reading, photography, the outdoors, science fiction, volleyball, bowling, frisbee golf, backrubbing and hugging. His present on-campus activities include an active involvement in Inter-Varsity Christian Fellowship and in the U.C.L.A. Concert Choir. John was a Teaching Assistant for CS10 in 1983-84 and a Teaching Associate for PIC10A and PIC10B from 1984-87. He is presently an Research Assistant for the Distributed Machine Intelligence Laboratory headed by Professor Vidal. His Ph.D. interests include Graphics, Computational Geometry, Parallel Processing, Theory, Vision, and Artificial Neural Networks.

Yadran Eterovic was born in Santiago, Chile in 1958. He received the degree of Electrical Engineer from the Catholic University of Chile in 1982 and the M.Sc. in Computer Science from UCLA in 1985. Between 1985 and 1987 he served as Assistant Professor of Computer Science and Assistant Director for Student Affairs in the School of Engineering of the Catholic University of Chile. He came back to UCLA last September to begin his Ph.D. His main interests are in the area of formal specification and validation of software systems.

Bob Felderman was born in Chicago, Illinois in 1962. He graduated Magna Cum Laude from Princeton University in 1984 with a double major in Electrical Engineering & Computer Science and Systems Engineering. After spending a year at Hughes Aircraft Company, he returned to the good life of academia and completed his Master's degree at UCLA in 1986. Currently he is pursuing a Ph.D. specializing in distributed systems with Professor Kleinrock. In his spare time he can be found in the great outdoors usually with frisbee in hand.

Dagan Feng is a Ph. D. candidate in the Computer Science Department at UCLA, with a specialization in Biocybernetics, a subfield of Scientific Computing. In addition to his formal training in Computer Science, he received a MS degree in EECS (Control Theory), at Shanghai Jiao Tong University in 1982, and a second MS degree in Biocybernetics, at UCLA in 1984. His dissertation advisor is Prof. Joseph J. DiStefano, III. The emphasis of his dissertation research is modeling theory and algorithms, with applications to biomedical systems. This research led to his winning the Ralph and Marjorie Crump Prize for excellence in Medical Engineering at UCLA in 1987. In addition to this area, his training and research interests also include other topics in Computer Science, such as robotics, controls, automation, CAD/CAM, etc. He has been a senior teaching assistant in the Computer Science Department at UCLA for the past three years. Last year, he was the head TA in charge of the Digital System Laboratory. This year, he is the recipient of the Mary Leonora Schulte Memorial Fellowship. He is a member of IEEE (Computer Society; Control Systems Society; System, man & Cybernetics Society; Eng. in Medicine & Biology Society; Acoustics, Speech & Signal Processing Society.), ACM (SIGBIO), CESASC.

Brett Fleisch was born in Hartford, Connecticut and received the B.A. degree in Computer Science at the University of Rochester. At Rochester, Fleisch participated in the development of RIG, and AMPS, two distributed operating systems. During the summer of 1981, after graduating, he visited Carnegie-Mellon University as a research programmer in the Department of Computer Science; at CMU he participated in the SPICE/DSN projects. From 9/82-5/83 Fleisch completed the M.S. degree in Computer Science at Columbia University. He joined the UCLA computer science department in September '83 where he is working towards the Ph.D. degree. Fleisch is currently a research assistant in the LOCUS group and one of Dr. Popek's advisees. In the past Fleisch has served as a consultant and summer employee to Xerox Corporation's, Wester Research Center, and IBM Corporation's Thomas J. Watson Research Center. Previously he served as a teaching assistant in the UCLA Computer Science Department and was involved in part time association with LOCUS Computing Corporation. He has been a reviewer for ACM Computing Reviews, has served as referee for past symposiums on operating systems principles, Transactions on Computer Systems, ACM Computing Surveys, and is member of the ACM and IEEE Computer Society.

William P. ("Bill") Fornaciari, Jr., was born in 1948 in Pasadena, Calif. He received his B.A. in 1970 from Wesleyan University with honors in chemistry, and his S.M., in 1971 from the University of Chicago in physical chemistry. The M.S. in applied physics was received in 1974 from the California Institute of Technology. From 1976 to 1982 he worked as a high school teacher of mathematics and programming. In 1981 he became an applications programmer with Bell & Howell Corp. During this time he resumed his teaching career at Cal State as an instructor of chemistry and computer science. The CS courses taught included introductory and advanced programming, programming language concepts and data structures. The M.S. in Computer Science was awarded in spring of '86 - the subject of the thesis was: "An Outline Editor." Bill has been a teaching associate for CS 174 and PC 10. Interests include software engineering, simulations & modeling, compilers, data organization, and computer science education. Bill is a member of the ACM.

Greg Frazier is completing his second year at UCLA. He is from Ann Arbor, MI, and received his B.S. in Computer Science and Engineering from MIT. Greg is majoring in Architecture, with minors in Networking and VLSI.

Maria Fuenmayor was born in Venezuela. She received the B.S. and Master Degrees in Computer Science from Universidad Simon Bolivar, Caracas, in 1980 and 1983 respectively. Her areas of interest are Cognitive Modeling, Natural Language Processing and Logic Programming. Other interests include music, good films, traveling and handcraft.

Hector Geffner was born in Buenos Aires, Argentina, twenty eight years ago. He obtained his B.Sc. in EE at the Universidad Simon Bolivar in Caracas, Venezuela, and his M.Sc. in Systems Science at UCLA. He is currently pursuing his PhD in AI under the supervision of J.Pearl. His interests in AI, which diminish with time, are essentially in the area of formal approaches to commonsense reasoning. He is also interested in general issues about science, technology and the third world.

Arthur Goldberg was born in Boston Mass. in 1955. He graduated Beacon Nursery school with a minor in jungle gym in 1961. 16 years of schooling later he graduated Harvard college with an AB in Astrophysics, magna cum laude. Art entered UCLA at the beginning of this decade. He received his Master's degree in December 1984 and is now working on finishing his dostorate. At UCLA he has worked as an RA, a TA, and taught extension classes. In addition he has worked at Rand, Silogic, and IBM research. Currently he is supported by an IBM doctoral fellowship. Art's research interests include load management of parallel programs, distributed operating systems, distributed system fault-tolerance, and discovering great oriental restaurants. His advisor is Professor David Jefferson.

Moises Goldszmidt was born in Caracas, Venezuela. He studied Electronic Engineering at the Universidad Simon Bolivar (Venezuela), where he got his degree in 1983 with a Cum Laude honor mention. In Fall 1985, Moises joined the University of California at Santa Barbara in pursue of a Ms degree in Electrical and Computer Engineering. There, he worked both as a Teaching Assistant and Research Assistant for the duration of his studies. Moises just started his PhD program at UCLA. His interests include: Artificial Intelligence, knowledge representation, common-sense reasoning and music.

Jim Goodwin: physicist in mid-life crisis. If Jim has his way, all the Star Trek episodes about thinking computers will come true. Academic interests: neural networks, parallel processing. Other favorite things: table tennis, far-out modern music, historians of medieval Japan.

Dan Greening is a Ph.D. student in hardware architecture, exploring the behavior of data flow hardware and software systems. He has done research work in the fields of operating systems, performance modeling, discrete mathematics, VLSI design, compiler construction, graphics, and networks. Dan has been employed in the field of computer science, mostly in operating systems research and development, for 13 years. He was a teaching assistant in Operating Systems (3 semesters) and Data Structures (1 semester) at the University of Michigan. He was a Project Manager for the Operating Systems Development Group at Contel Business Systems. He was a Visiting Research Associate at the MIT Laboratory for Computer Science, studying data flow architectures. Dan has had a life outside computer science as well. In 1985-1986 he was elected

President of the University of California Student Association, the system-wide student goverment. There he coordinated a statewide Board of Directors, a Sacramento lobbying organization, and a $200,000 budget. He also was elected UCLA Graduate Student Association Vice-President, External Affairs. He has been a member of the UCLA Student Conduct Committee since 1986. Dan backpacks in wild places, wanders through muddy caves, eschews television, revels in modern art museums, thinks about existential philosophy and history, and forgets the punch-lines in jokes.

Richard Guy was born in Takoma Park, Maryland in 1959. He graduated from Loma Linda University in 1981 with honors, receiving a B.S. in Computer Science. He has been a research assistant with the DARPA/LOCUS Distributed Systems Research Group since 1981, also serving as a computing facilities manager from 1981-1985. He received his M.S. in 1987, with the thesis title "A Replicated Filesystem Design for a Distributed Unix System," and is currently pursuing doctoral research in replicated filesystems and databases. Completion of the Ph.D. is expected in late 1989. Additional interests include distributed systems architecture, network protocols, and computer systems theory.

Lars Hagen was born in L.A. in the year 1963, and grew up all over the world (primarily Michigan and Norway). He attended several different schools, and eventually acquired a B.S. in Engineering Computer Science from C.S.U. Long Beach. Presently Lars is studying for his M.S. under Professor Parker's guidance with a departmental fellowship. He hopes to transfer to the Ph.D. program soon and get involved in a thesis project. His interests are Formal modeling, Logic programming, cultural events and fun people.

Othar Hansson is descended from Magnus the Barefoot, Ivar of the Long-Reach, Vemund Word-Master, Harald War-Tooth and Hraerek Ring-Scatterer, and was born just outside of London in 1965. In January, 1986, he received his A.B. in English Literature and Computer Science from Columbia, and the following autumn entered UCLA as a Research Assistant to Prof. Richard Korf. His interests include programming languages, algorithms, problem-solving and learning, as well as language(s), literature(s), translation(s) and education. As he hopes to one day be president of the National League, he plans to enter academia upon earning his Ph.D.

Born in Louisville, Kentucky but raised in Ann Arbor, Michigan **Jack Hodges** has always been a Wolverine fan. Nevertheless, upon graduation from the University of Michigan with an MSE in aerospace engineering (1978) he made the pilgrimage to California. For what seemed to be eons he worked as an engineer at a local aerospace firm. He did a lot of hang gliding. Eventually Jack saw the light and returned to the academic arena in 1983, this time masquerading as a computer science student while secretly planning to steal precious secrets back to the engineering field. Jack has been studying AI and, in particular, the representation of simple mechanical devices and how people with no technical training can be inventive.

Miquel Huguet received his undergraduate *Licenciado* degree from the Computer Science School of the Politechnic University at Barcelona in 1979, graduating with honors. From 1980 to 1982 he was teaching and doing research at the Computer Architecture Department of that School. In 1982 he was awarded a FULBRIGHT/MEC scholarship and came to UCLA Computer Science Department. Since 1984 he has been teaching courses on Operating Systems and Digital Systems. He received his M.S. in Computer Science degree in 1985. He is currently working towards his Ph.D. His general research interests are in the field of computer architecture with special emphasis in instruction set design and evaluation, optimizing compilers and code generation, and architectural support for operating systems.

Richard Huntsinger is currently recovering from the culture shock of migrating from rural northern California to metropolitan Los Angeles, timesharing between doctoral studies and scuba dives. His background includes studies in chemical engineering at UCD (Davis), mathematics at UC (Calgary), computer graphics and simulation at CSUC (Chico), and music at UCSB (Santa Barbara), with professional appearances at IBM in San Jose and a wind turbine farm in Livermore. He has lectured on programming languages and graphics as visiting faculty at UC, CSUC, and UCSB. Now, at UCLA, he is completing a dissertation involving constraint satisfaction algorithms and their applications to music composition, animation, and simulation.

Andy Yuan-Maw Hwang was born in Taiwan, ROC, in 1957. He received his B.S. degree in industrial engineering from National Tsing Hua University, Taiwan, in 1979. He went to Northwestern University in Sept. 1982 and received an M.S. in industrial engineering and operations research next year. Two months later, he joined University of Illinois, Chicago and finished his second M.S. degree in computer science. He came to UCLA for his Ph.D. degree in Sept. 1985. His interests are distributed database systems, network modeling, and tennis.

Arthur Ieumwananonthachai was born in 1967 at Ann Arbor, Michigan. He was raised in Thailand and received a BS in Electrical Engineering and BS in Computer Science from University of Washington (Seattle, WA) in 1986. For one year, he worked at Hewlett-Packard, Lake Stevens Instrument Division (Everett, WA) in Productivity Section (R&D) for workstations and LAN support. He is now working for a MS with plans to continue on to the PhD program. His special interest is in computer network and architecture but really wants to learn about everything that involve computers.

Alan Ishigo was born and raised in Gardena, CA. He received a B.S. in Math/Computer Science from UCLA in '85. At this time, Alan decided to work for a couple of years before going back to school and is currently working in the VLSI Design Automation section at Xerox Corporation in El Segundo. His academic interests include looking into expert systems as applied to design automation under Prof. McNamee. He also enjoys basketball, skiing, bicycling, bowling ... basically anything semi-active. He also detests commuting between home, work and school -- there's gotta be a better way!

Jim Kan was born in Taipei, Taiwan. Later, he attended the University of California, Irvine and received B.S. degrees in Electrical Engineering and in Information & Computer Science. He is currently pursuing a Master's degree in artificial intelligence and is interested in natural language processing and cognitive science. He enjoys annoying office mates by listening to his walkman and singing Lou Reed songs off-key.

Ted Kim was born in Boston, MA in 1962 and grew up in Haverford, PA. After recieving his SB in Computer Science and Engineering from MIT in 1984, he came to UCLA for graduate studies. Currently, he is in the MS program in the field of distributed systems and is a PGRE for the LOCUS research group after having been a bounty hunter and a TA for CS141. He also ekes out a modest living as the food and wine editor of *CyberPunk* magazine. Ted's interests include games, sailing and strategic studies.

Willard Korfhage was born on Aug. 22, 1960. Childhood interestest in electronics resulted in his designing his first computer in high school, and later resulted in several degrees in computer science (BSE in EECS from Princeton in 1982, MS in CS from UCLA in 1985). His current dissertation research is a theoretical analysis of networks of workstations (such as the BBL project), and he expects to complete his dissertation within a year. Willard is also a sailing instructor for UCLA, and spends most weekends sailing either catamarans or UCLA's 50 foot keelboat, with occasional windsurfing thrown in for variety. He kills off his remaining free time with photographic work, concerts, and cooking.

Milan Kovacevic was born in 1960 in Belgrade, Yugoslavia. He received his diploma in Electrical Engineering in 1985 from the Faculty of Electrical Engineering, University of Belgrade. His major was Electronics, but he also completed the Computer Engineering program. In 1987 he received his M.Sc degree in Computer Engineering from the same university. From 1985-1987 he was a research assistant at the Department of Electronics and Computer Engineering, University of Belgrade, where he worked on the development of system software for a fault-tolerant multiprocessor system. Since 1987 he has been a Ph.D. student at the UCLA, Computer Science Department, working as a RA for professor Milos Ercegovac. His research interests are computer architecture, distributed systems, operating systems and modeling. In his spare time he prefers playing table tennis and watching movies. He is currently the UCLA table tennis champion.

Lance Kurisaki was born in Honolulu, Hawaii. He graduated from UC Berkeley with a B.S. degree in Electrical Engineering and Computer Science in 1983. He received his Master's degree in 1984 from Stanford University in Computer Science, and is now working on his Ph.D. His academic interests include computer architecture and programming languages.

Koenraad Lecot is currently enrolled in the Ph.D. program and plans to finish in June, 1988. His (academic) fields of interest are databases, expert systems and logic programming. His dissertation research involves inexact reasoning techniques in knowledge-based systems. Koenraad got his B.S. in computer science from the University of Brussels, Belgium and his M.S. from UCLA in 1984.

Geunbae Lee was born in Chung-na, Korea. He studied Computer Engineering at Dept. of Computer Eng., Seoul National Univ. of Seoul (Korea) and graduated as a honor student in 1984. He finished his master at same univeristy , studying distributed system and computer network, in 1986. He was a Unix system manager in the computer center at SNU between 1984 -1986. He joined UCLA Ph.D. program in the Fall of 1986. He is RA at AI lab now,working with Prof. Dyer and Flowers. His academic interests include: connectionist modeling of natural language understanding and general artificial intelligence fields.

Jeong-A Lee was born in Pusan, Korea. She received B.S. in computer engineering from Seoul National University, Korea in 1982. After working at Gold Star research institute, Korea as a research assistant for one year, she came to the United States to further her studies in computer science. She got M.S. in computer science from Indiana University, Bloomington in 1985. At Indiana, She was a TA for two years and found out that teaching could be an interesting profession for her. To achieve her goal, to become a professor, she has joined Ph.D. program in the field of system architecture since fall of 1985. She is currently CS152A TA.

Ting-Yu (Cliff) Leung was born in 1962 in Hong Kong. He studied Electronics and Computer Science at the Chinese University of Hong Kong, and graduated from the Department of Electronics in July 1984. He entered the University of Texas at Austin in the Fall of 1984. In August 1986, he received a Master's degree from the Department of Computer Science. Ting-Yu joined UCLA in the Fall of 1986 and is now working on his Ph.D. His academic interests include: database design, distributed database, and software automation.

Paul Chang-Hsin Lin began his life in 1962 in downtown Taipei. At age 13, he moved to Hawaii and started his journey in America. He received a B.S.E.E. from Rice University at Houston with minors in computer science and mathematical sciences. After his undergraduate study, he travelled north to work for Sperry in Minnesota for a summer. At the end of summer, he came to southern California to pursuit his M.S. in computer science at UCLA. At UCLA, he became a jack of all trades: TA for Robotics Lab and Computer Vision Lab, RA at Machine Perception Lab, kitechen chief, maintanence man, and cleanup crew at the Co-op. Currently, he is finishing his M.S. thesis in the area of computer vision under the guidance of Dr. Skrzypek and searching for a real job.

Tzung-I Lin was born in Taipei, Taiwan 1961. He received his B.S. degree in Electrical Engineering from National Taiwan University in 1984. Having served as a lieutenant in ROC Army for two years, he came to UCLA to pursue his M.S./Ph.D. degrees in 1986. His research interests include network modeling, interconnetion network, neuro-network and distributed systems. Leisure time interests include travel, movies, military technology, hiking, Chinese chess, reading, all kinds of sports and other bizarre experience.

Xinming Lin was born in Hunan, People's Republic of China, in 1962. He received his B.S. from the Changsha Institute of Technology, China, in 1982. He came to UCLA in 1983 and received his M.S. in Computer Science in 1985. He is now working on his Ph.D. dissertation in designing a new programming language for Scientific Computing. His academic interests include physical system simulation, programming languages, and parallel processing. He is a member of IEEE Computer Society, ACM, SCS, and SIAM. He always enjoys reading, thinking, and physical exercising.

Robert Lindell received his bachelor's degree in Math - Computer Science from UCLA in 1984. He received a Master's Degree in 1986 for work in the area of biological modeling. He is currently in the PhD program at UCLA. Robert has worked on the LOCUS research project since 1981 and is currently working on the Tangram research project.

Edward Lor is a Ph.D. student majoring in Programming Systems. He got his B.S. from the University of Maryland in 1980, his M.S. from U.C.L.A. in 1982. He is currently working in the SARA/IDEAS group and has just finished his dissertation entitled "An Assistant for Requirment Driven System Design". His major areas are software engineering and programming languages. He is particularly interested in automatic programming and requirement analysis.

Stephen Lui was born in Los Angeles, CA and has resided in Southern California all his life. He studied Computer Science and Engineering at UCLA and earned his BS degree from the School of Engineering and Applied Science in June 1986. Stephen continued his Computer Science education by entering the Master's program in the Fall of 1986 and expects to complete this degree in the Spring of 1988. He currently works for Hughes Aircraft in Culver City as a Computer System Manager. His academic interests include: computer architecture computer graphics, and business information systems and is currently involved in the project titled "Performance of Dataflow Architectures" under his faculty advisor, Tomas Lang.

Matthew Merzbacher is from Chapel Hill, North Carolina. He received a pair of degrees from Brown University before coming to UCLA to study commuter science. After all, what better place to study commuters than LA? He is working towards his Ph.D. in distributed database managment. If you know anything about this subject, please drop a line - you'll be acknowledged in the dissertation.

Risto Miikkulainen is an AI student from Helsinki, Finland. He started out as a physics student in the Helsinki University of Technology, took a year off to get a B.A. degree from Southwestern University in Texas in 1984, and went back to HUT to finish his M.S. in systems analysis and operations research in 1986. After a 6 month trip around the world he joined the Ph.D. program in UCLA in the fall of 1986. Currently he is working as an RA in the Ailab, doing research in connectionist models of natural language processing.

Jose Miro-Julia was born in Cleveland (Ohio) some day in 1961. When he was five he flew to Spain. He completed there an undergraduate degree in Physics, at the Universidad Complutense in Madrid by June 1984. That same year he started some graduate work at the Universitat de les Illes Balears in Palma de Mallorca. After his first year there he shifted from the Physics Dept. to the Maths Dept. He expects a Dr. in Science degree in Summer 1988, his research being about automated theorem provers using a four valued unconventional logic. He joined UCLA in Fall 1987, and, having decided to do yet another shift, went to the Computer Science Department. His major is computer architecture.

Jose Augusto Suruagy Monteiro was born in 1957 in Recife, Brazil. He graduated in Electrical Engineering at the Universidade Federal de Pernambuco (Brazil) in 1979, and received a Master's degree in Electrical Engineering (Digital Systems) at the Universidade de Sao Paulo (Brazil) in 1982. Since 1983 he was an assistant professor at the Computer Science Department of the Universidade Federal de Pernambuco. Jose joined UCLA in the Fall of 1985 where he has been working for his Ph.D. His academic interests include: computer network protocols, distributed systems, and programming languages.

Jaime H. Moreno is a doctoral student at UCLA. He received an Electrical Engineering degree from Universidad de Concepci´{o}n (Chile) in 1978, and a M.S. in Computer Science from UCLA in 1985. He was with the faculty of the Electrical Engineering Department at Universidad de Concepcion from 1978 to 1983 and has been a Teaching Assistant at UCLA from 1984 to 1987. He is currently the recipient of an IBM Graduate Fellowship and previously received a UNDP fellowship to pursue his Master's studies at UCLA. His areas of interest are computer architecture, digital systems for special purpose applications, and VLSI design.

Sergio Mujica was born in Santiago, Chile and graduated from Pontificia Universidad Catolica de Chile. He received a M.Sc. degree in Computer Science from UCLA in 1978. Sergio was admitted to the PhD program in Computer Science in 1985. He is now leading the SARA/Ideas group to the implementation of a new generation of the SARA tools for support of design, and his research activity is focused in the *collaborative design* of computer systems.

Valeriy Iliev Nenov was born in Sofia, Bulgaria in 1953. He attended Sofia State University, Department of Physics during 1972/73. In 1979 he received his Engineer's degree (M.S.) in Physical Electronics from the Prague Polytechnic University in Czechoslovakia, Nuclear Science and Engineering Department. In Prague he was also a research assistant at the Brain Research Institute, Czechoslovakian Academy of Sciences. Back In Bulgaria in 1979 Mr. Nenov worked for four years as a service engineer and senior programmer at the GC/MS Lab, Institute of Organic Chemistry at the Bulgarian Academy of Sciences. During this period he was twice a short term visiting research assistant at the Department of Structural Chemistry, Ruhr University in Bochum, West Germany, and the Japanese Electron Optical Laboratories (JEOL) in Tokyo. In the Fall 1983 Mr. Nenov was admitted in the Interdepartmental Neuroscience Ph.D. Program at UCLA. His dissertation research in neuroscience advised by Professor Eric Halgren is concerned with the study of human memory through Positron Emission Tomographic methods. In the Spring of 1986 Valeriy was admitted as a graduate student in the Computer Science Department at UCLA with a major in AI. At CSD he was involved with the development of the MOZAK project concerned with computer comprehension of neuroscience abstracts and supported by Hughes Aircraft Corporation. Currently his main research interest is in connectionist modeling of associative functions in humans. His CS dissertation advisor is Professor Michael Dyer. Leisure time interests include foreign languages, flying and diving.

Tom Page received his B.S. from Duke and his M.S. in Computer Science from UCLA. He is a former member of the Locus Distributed Operating System project and is now working on the Tangram project with Professors Muntz, Parker, and Popek. Tom is working on his disertation on transparent interfaces between logic programming languages and distributed databases.

Alex Pang was born in 1959 in the world's banana capital which is Davao City, Philippines. He studied Industrial Engineering at the University of the Philippines and graduated with magna cum laude honors. Then he saw a UCLA commercial in one of the football reruns and decided to come over and meet some of the girls from the cheering team, ending up in the Computer Science Department. Now, Alex has an MS in CS doing motion prediction based on satellite image processing and is back in school for his PhD. This time, his attention has shifted to computer graphics and animation as well as parallel processing. While at school, Alex maintains his financial stability and social life by TAing for E10, CS20 and more recently CS 11. He still keeps in touch with nature by camping out and windsurfing. Occasionally, he reverts to his old hobby of counting ants.

Dorab Patel has been around this department so long that he can remember what life was like before LOCUS. In the distant past he received a B. Tech in EE from IIT - Bombay and an M.S. in CS from UCLA. He has hung around the SARA/IDEAS and FP-VLSI groups for far longer than he can recall. In the last year, his major accomplishments, in the face of tremendous odds, include consuming five kilos of fine Belgium chocolate, perfecting a kulfi recipe, and porting ChezScheme and GNU EMACS to UCLA. During the time our machines are in *partition*, *merge* or *wait*, he researches functional languages and their application to VLSI synthesis.

Jody Paul was born in Brooklyn, New York City, NY. He holds a B.S. degree in Mathematics-Computer Science and both M.S. and Ph.D. degrees in Computer Science from UCLA. His Master's focused on practical applications of artificial intelligence and computer technology (a program in cooperation with the UCLA Graduate School of Management). His doctoral research concerns a new method for producing software systems that can explain themselves to users: "Self-Revealing Software". His thesis describes the methodology and demonstrates its application in the development of an expert system that evaluates asbestos product-liability claims. Jody currently is a consulting computer scientist and knowledge engineer for the RAND Corporation. He also teaches courses in natural language processing for UCLA Extension and lectures on AI and expert systems for universities and industry. Jody is a member of the American Federation of Musicians (Local 47), holds USYRU certification as a dinghy and keelboat sailing instructor, and teaches sailing for the UCLA Sailing Program.

Michael Pazzani was born in 1958 in New York City. In 1980, after receiving his B.S. and M.S. in computer science from the University of Connecticut, he went to work at the Mitre Corporation. Eventually, he became group leader of the Artificial Intelligence Research group, and then he left

to pursue his Ph.D. at UCLA. His interests include learning and natural language understanding. His hobbies are bird watching and professional wrestling. Michael is supported by the UCLA/RAND Artificial Intelligence Fellowship.

Joseph Pemberton received his B.S. in Electrical Engineering from the Massachusetts Institute of Technology (MIT) in June 1984. He then worked as a staff engineer for the Charles Stark Draper Lab from June 1984 until August 1985. Joe entered UCLA in the Fall of 1985. He is currently working towards his Master's degree and is employed as a research assistant in the Distributed Machine Intelligence Group. His academic interests include: artificial intelligence (specifically intelligent harware architectures and machine learning paridigms), pattern recognition, computer architecture, and brain modeling. His other interests include music and soccer. He is currently serving as a graduate student representative to the computer science faculty (csrep) and as vice-president of the Engineering Graduate Students Association (EGSA).

Brad Pierce was born in South Bend, Indiana in 1963. He graduated twice from Indiana University-Bloomington in 1984 with the BS in Mathematical Sciences and the BA in Germanic Languages and a third time in 1986 with the MS in Computer Science. Like all graduates of Indiana University, he is a Scheme evangelist, and currently teaches Scheme as a TA at UCLA. An Artificial Intelligence major studying neural networks, he started working towards the Ph.D. in Fall 1986. His main interests are connectionist computing, coffee drinking, and Scheming.

Arturo Pizano was born in Mexico City in 1958. In 1980 he received a Bachelor's degree in Actuarial Sciences from the National Autonomous University of Mexico, where he also met his now wife Lourdes. After three years of working for the Mexican Treasury Department, and a large private corporation, he came to UCLA in 1982 on a scholarship from his alma mater. He obtained his Masters degree in 1985 and is currently working on his Ph.D. under the direction of Dr. Cardenas. Since 1984 he has also been a member of the Administrative Computing Staff of the School of Engineering. His research interests include pictorial database management, data modeling, and semantic integrity control. As a proud father of two children, Arturo and Constanza, he enjoys spending time with his family. He also plays golf and can be seen at every UCLA football or basketball game.

Curt Powley is a Ph.D. student in artificial intelligence doing research in parallel search. During his two hours of free time each week, he also works on keeping the right attitude toward life via cultural excursions to different Los Angeles microworlds, maintaining a sufficient cardio-vascular level (e.g., an occasional dawn surf), seeing movies, and reading literature.

Maria M. Pozzo received her M.S. degree in Computer Science from the University of Connecticut in 1981. She began working in the area of Computer Security for the Mitre Corporation and subsequently went to work for Honeywell in 1984. At Honeywell, Maria worked as a systems programmer and was responsible for a portion of the Multics security kernel. She began work towards her Ph.D. in 1985 and during her first year worked primarily on the computer virus problem, publishing several papers with her advisor. Maria's main area of interest is computer security with a particular focus on programs that contain Trojan horses, computer viruses, and other programs that perform malicious activity. She is also interested in computer privacy, legal issues, and the social impact of computing.

T. M. Ravi was born in Urbana, Illinois and raised in Kanpur, India. He received his B.Tech in Electrical Engineering from the Indian Institute of Technology, Kanpur and his M.S. in Computer Science from UCLA. His interests are in Distributed Systems, specifically - high speed architectures, distributed algorithms, distributed operating systems and data flow techniques. He is currently studying "Process Migration in Large-Scale Distributed Systems". He is a member of the team involved in the implementation of dynamic load management on the hypercube. In the past he has worked at the IBM Thomas J. Watson Research Center and currently is a recepient of a IBM Graduate Fellowship. He is passionately committed to Romanticism, and has a strong appreciation for the mystery and beauty of nature. He is interested in political analysis, issues concerned the Supreme Court, the crisis in the American educational system and poetry.

John Reeves received his B.S. in Computer Science from the University of California, Santa Barbara, in 1980. He entered UCLA in 1983, received his M.S. in 1986, and is currently pursuing

a Ph.D. in artificial intelligence. An R.A. in the Artificial Intelligence Laboratory, his research interests are belief conflict, irony, and cognitive models of language generation, including news stories and conversation. He lives with his wife and a cat in Redondo Beach.

Jason Rosenberg was born in San Francisco and grew up in the Bay Area. He graduated with a degree in Electrical Engineering and Computer Sciences from the University of California, Berkeley in 1987. He intends to study computer architectures, with emphasis on massively parallel systems. In addition, he is interested in exploring facets of artificial intelligence, such natural language processing and perception.

Rony Ross was born and raised in Tel Aviv, Israel. She received the B.Sc degree in Mathematics (cum laude) from Tel-Aviv University in 1972; the M.SC degree in Computer Science (cum laude) from the Feinberg Graduate School of the Weizmann Institute of Science, Rehovot, Israel in 1976; and the MBA degree in Business Administration from the Graduate Scool of Management, Tel Aviv University, in 1980 (cum laude). From 1971 to 1980 she was a teaching assistant at the Tel-Aviv University, Computer Science Department; the courses she taught included Computer Organization, Introduction to Computer Science, Introduction to Programming (Pascal, Fortran), Switching Theory and Finite Automata. From 1975 to 1980 she was employed by Mini-Systems Computers Ltd. as Systems Analyst and Real-time Programmer, working for Sci-Tex Corp. From 1980 till 1983 she was Manager of Data processing and Information Systems Department of Kitans Ltd. From 1983 till 1986 she worked for Contahal Ltd., Israel's second larest software house: first in charge of New Business Development and later as Manager of the Artificial Intelligence and Expert Systems Department. She joined UCLA in September 1986 where she is working towars a Ph.D. degree. Her extra-curricular activities include raising two kids (Guy and Galy), a husband (Gideon), reading, skiing, TV soaps and Israeli. folk dances.

Rony Shaul Sawdayi was born in Tehran, Iran, went to high school in Ramat-Gan, Israel, and entered the United States in 1982. He graduated Summa Cum Laude from UCLA in December 1986 in Computer Science & Engineering. He was the senior recipient of Golden Key Scholarship Award, and a member of Tau Beta Pi. During his senior year he worked as Engineering Aide under Professor Michel Melkanoff. He joined UCLA's M.S. program in Computer Network Modeling & Analysis in September 1987 as a Department Fellowship recipient. Among his extra-curricular interests are traveling, dancing, and skiing.

Eve Schooler was born in New York in 1961. She received her B.S. in Computer Science from Yale University in 1983. After a two year stint working at Apollo Computer, Inc., she was inspired to return to school to pursue her M.S. degree. Specializing in distributed systems, she is currently a research assistant to Professor Leonard Kleinrock and has been involved in the Benevolent Bandit Laboratory project, a testbed for distributed algorithms. Her thesis is concerned with distributed debugging in a loosely-coupled processing system. In addition to her interest in scientific computing, she is also extremely interested in mixing technology, education, and the arts. In her spare time, she is an avid music enthusiast.

Werner Schuetz was born in Vienna, Austria, in 1961. He studied Computer Science at the Technical University of Vienna where he worked with Professor Kopetz while doing research for his thesis: "A Reliable LAN for Real-Time Applications". He graduated and received the academic degree "Diplomingenieur" in June 1986. After that he decided to leave the cold European weather behind. He received a Fulbright scholarship and came to UCLA where he is now working for his Master's degree. Currently, he is also working as a research assistant with Professor Avizienis. Among his academic interests are: fault tolerant systems, computer networks & communications, distributed systems, and distributed processing. Other things he likes include skiing, mountaineering, books, movies, good food, and good friends.

Marti (Reisman) Sheldon was born in 1961 in Miami, Florida. She studied Computer Science at Cornell University in Ithaca, NY and graduated "with distinction" in 1983. She received the Bell Laboratories Award for top junior in Computer Science in 1982. She worked at Hughes Aircraft Company in the Cornell Engineering Co-op program Fall of 1981 and Summer of 1982. She now works at Hughes on a part-time work-study fellowship. She is now working toward her Master's degree. Her academic interests include: systems engineering, structural design, and database management.

Chien-Chung Shen was born in Taiwan, Republic of China, on July 15, 1960. He received the B.S. and M.S. degrees in Computer Science from National Chiao Tung University, Hsinchu, Taiwan, in 1982 and 1984, respectively. His research interests include data base, distributed system, and AI. He also like jogging and collecting stamps.

Gene Sheppard was born in Decatur, Illinois. He grew up in Illinois and Indiana. Tiring of winter and increasingly interested in computers, Gene moved to Texas in 1983, and, in 1987, received a B.A. in Computer Science from the University of Texas at Austin, graduating with highest honors. His work background includes experience in the railroad and construction industries, as well as time spent as a programmer. He is currently enrolled in the joint MBA/MSCS program at UCLA. Gene's academic interests are Data Management Systems and Information Architectures. His hobbies include computers, science fiction, running, and bicycling.

Eric Chi-Man Sit was born in Hong Kong in 1960. He studied Electrical Engineering at Polytechnic Institute of New York. He received his Bachelor's degree in June 1983. Eric then joined UCLA in the Fall of that year. He has been working as research assistant since Dec. 1984. Now he is working on his Ph. D. His research interests include: modeling of distributed systems, distributed processings and distributed databases.

David Smallberg was born and raised in Los Angeles. He received the B.S. in Mathematics from the California Institute of Technology, and the M.S. in Computer Science from UCLA. His work on behalf of undergraduate education earned him the Department's Distinguished Service award in 1985 and a UCLA Outstanding Teaching award in 1986. He has taught both at UCLA and in industry, and has co-authored a text on programming a digital signal processor. He is interested in protocol specification and verification, and parallel algorithms. He has a personal interest in software psychology — he believes that he's a spreadsheet program. His other interests include skill-gimmick and enigma ralleys, recreational linguistics, and Asian cuisines.

Basuki N. Soetarman is pursuing a PhD in Computer Science majoring in Software Systems. He received an M.D. degree from the School of Medicine at the University of Indonesia, and an M.S. in Computer Science from UCLA. His main interest are in programming languages, data base and knowledge base system, logic programming, CAD/CAM, simulation, and medicine. He lives in sunny Santa Monica, and he loves his country, Indonesia, very much.

Scott Spetka received his B.S. degree from Denison University in 1977. He proceeded to El Salvador to visit his brother and enjoy a much needed vacation. After meeting and marrying his wife Rosemary, he looked for and found work there. Inspired by the IBM 370/115, he developed an interest in operating systems. That interest became an interest in distributed operating systems and communication networks when he left his job as Technical Advisor to the Computer Center at the Salvadorean Social Security Institute to accept a similar job at the National Telecommunications Administration. After serving as a Teaching Assistant for one year at UCLA, Scott joined the Locus Research group. His research is concerned with the Locus distributed operating systems and distributed database management systems. Scott is an avid UCLA football and basketball fan. He is also the father of one, and soon to be two. If these two interests are not enough, he is involved in student government through the graduate student association. He has served four terms as President of the Engineering Graduate Student Association and three terms as an Engineering representative to the GSA FORUM. He also served last year on the ASUCLA Communications Board and is currently a member of the ASUCLA Board of Control.

Mike Stiber started life in 1962 in Philadelphia, Pennsylvania, and has been working his way west ever since. He graduated from Washington University in Saint Louis with two Bachelor of Science degrees -- one in Computer Science and one in Electrical Engineering. Next came a short interlude in the Netherlands growing computerized tulips for Philips, and then two years of Dallas Cowboy games with Texas Instruments. He will soon receive his MS in Computer Science at UCLA, and will continue for a PhD, focusing on the application of massively parallel architectures to general-purpose visual tasks.

Paul E. Thorpe was born in Kingston, Jamaica, on October 4, 1964. He graduated Magna cum Laude in 1986 from Loma Linda University, Riverside, CA, with a B.S. in Computer Science and an A.S. in Engineering. He is currently working on his Masters at UCLA with the intention of continuing on to the P.hD. His areas of interest include distributed systems, operating systems, programming languages, and compilers. He has received a Fellowship from Pacific Telesis.

E. Robert Tisdale is a former resident of Minnesota, Washington, Alaska and New Mexico. He obtained a B.S. in Physics from the University of Washington in 1979, worked as a Petroleum Engineer from 1980 to 1985 and obtained a second B.S. in Computer Science from San Francisco State University in 1987. He came to the University of California at Los Angeles in the Fall of 1987 to pursue the M.S in Computer Science with a major in Scientific Computing. He is a teaching assistant for the Introductory Digital Circuits Laboratory. He is a private pilot and an advanced open water diver.

Jay Tobias, formally known as John C. Tobias II, received his Bachelor of Science degree in Math/Computer Science from UCLA in June, 1985. He gained valuable and interesting experience during his ensuing employment at Hewlet-Packard as a software development engineer and at the Center for Music Experiment at UC San Diego as an assistant researcher and systems programmer. Having entered the graduate program in the fall quarter of 1986, Jay works as a teaching assistant for CS 111, Introduction to Operating Systems, and pursues research in the area of realtime computation on massively distributed systems. He was a member of the UCLA programming team that placed second at the regional contest in November, 1986. Other interests include computer and non-computer music, athletics, and a Jungian approach to individuation.

Peter Trajmar is a California native, born in nearby Pasadena. He received his B.S. in electrical engineering and computer science from the University of California at Berkeley in May of 1987. He is currently working towards his M.S. degree with the area of specialization of computer architecture, but also with a keen interest in software, especially relating to ease of use. Non academic interests include football, basketball, and volleyball. He is also a Mac enthusiast.

Marc Tremblay was born in Quebec, Canada in 1961. He received his B.S. in Physics Engineering from Laval University in June 1984, and his MS in Computer Science from UCLA in June 1985. He is now pursuing his Ph.D on a scholarship from the National Research Council of Canada. His academic interests are in the field of Computer Architecture with a special emphasis on the design of processors, their VLSI implementation, and their fault-tolerance capabilities. He has been a Teacher Assistant for the class CS-151A (Comp. Arch.) for a couple of years, and according to him, he likes it almost as much as playing tennis. Marc's activities and interests include gymnastics (former member of the junior national team), tennis, windsurfing, skiing, hockey, golf, movies, science-fiction books, plays; he loves to travel to exotic places.

Paul Kangguo Tu was born in Ithaca, NY. He went to China at the age of 6, and grew up in Beijing (Peking), China. He received his B.S. in mechanical engineering from the Wuhan Institute of Iron and Steel Technology in 1976. He came to the U.S. in 1979 and received his M.S. in computer science from the University of Wisconsin-Madison in May, 1981. He is currently working for his Ph.D in computer architecture. His main interests include computer arithmetic and algorithms, parallel processing, systolic array architecture, and operating systems.

Tella Vijayakumar was born in Hyderabad, India in 1965. He received his B.Tech in Computer Science from the Indian Institute of Technology at Madras in July 1986 and is now working for M.S. in Computer Science. He is currently a TA for CS141. His areas of interest are computer networks, distributed processing and artificial intelligence.

Rich Wales was born in 1952 in San Mateo, California. He graduated from Stanford University in 1975 with a double major in Mathematics and Music. After receiving an M.A. in Music from UCLA (with an emphasis in music education) in 1977, he came to the Computer Science Department and earned his M.S. in 1979. Rich was a Teaching Assistant and Teaching Associate from 1977 to 1980, during which time he taught sections of the introductory programming course. Since 1980, he has been a programmer/analyst for the UCLA Computer Science Department. He plans to complete his Ph.D. in distributed software management during 1988. In addition to his research area, Rich's other areas of interest include networking issues and electronic mail systems.

Ping-Hann Wang was born in Taiwan, R.O.C. in 1962. He received his BS degree in electrical engineering from National Taiwan University in June 1984. He joined the China Air Force as an officer for two years. After that, he joined UCLA in the Fall of 1986. He is currently working on his MS. His academic interests include: computer architecture, distributed processing, cache memories, and VLSI CAD. Additionally, he likes classical music, basketball, swimming and traveling.

Yih-Jih Alan Wang was born in Taipei, Taiwan, ROC. He received his B.S. degree in Electrical Engineering from National Taiwan University in 1980. He went to University of California, Santa Barbara in September 1982, and received his M.S. degree in June 1983. He then worked for Burroughs Corp. for two years and joint UCLA in September 1985. His major interest is nature language processing and neural networks.

David Wine entered the Ph.D. program in Computer Science in January, 1988. He is acting as a teaching assistant for CS161, Fundamentals of Artificial Intelligence, and his research interests are in artificial neural networks as they are applied to artificial intelligence. David attended the University of Pennsylvania where in 1984 he received a BSE in Computer Science and Engineering from the School of Engineering and Applied Sciences and a BA in Psychology from the College of Arts and Sciences. While in Philadelphia, David took a year out of school to be a partner in a start up software company. David has also done work in the color vision lab of Psychology professor Dr. Edward Pugh, and in the neurophysiology lab of Pharmacology professor Dr. Solomon Erulkar. For the past several years, David has been working for IBM in New York in the area of semiconductor yield prediction and analysis, and has been a reviewer for the IEEE International Test Conference. David was also responsible for promoting and coordinating expert system development in the Technical Services organizations of IBM East Fishkill. David's interests include indoor gardening, board sailing, contract bridge, and ultimate frisbee. His office is at 3526-I Boelter Hall, and he invites you to stop by.

Greg Wong is a native of Southern California, part of the third generation of his family to attend a UC school and of the second generation to attend UCLA. He received his BS in Math/Computer Science from UCLA in 1986 and is enjoying his second year as a graduate student, working toward an MS in Computer Science. His academic interests include computer graphics, and discrete event simulations. This is also his second year as a Teaching Assistant - he has been the TA for CS-11, CS-12, and CS-13 during both of those years. His personal interests include classical music, photography, traveling, simulation gaming, fishing and basketball. He is something of an amateur composer and have written a symphony and a concerto.

Winthrop Wu was born in Alliance, Ohio in 1962. He studied Computer Engineering at Case Western Reserve University and graduated from the Department of Computer Science & Engineering in May 1984 He has been a teaching assistant for CS152A and CS151B and is now a research assistant Presently, he is working on his Master's degree. His academic interests include: computer architecture, VLSI design, computer graphics

Samuel C. Yang has a BA in Biology from the University of Pennsylvania, and an MS in Computer Science from Rensselaer Polytechnic Institute, where his Master's Project earned him a place in Who's Who in Computer Graphics. Sam has worked for AT&T, IBM, and Xerox, where he still works part time developing workstation software. His academic interests currently include: neural networks, natural language processing, and pattern recognition.

TABLE I
PH.D. DEGREES AWARDED IN CALENDAR YEAR 1987

NAME	DISSERTATION TITLE	MAJOR FIELD	ADVISER	COMPLETION DATE
Belghith, Abdelfettah	Dynamic Routing for Real Data Transport in Mobile Packet Radio Networks	Modeling & Analysis	L. Kleinrock	March 1987
Campbell, Michael Lee	Data Flow Graphs as a Model of Parallel Complexity	Theory	S. Greibach	March 1987
Chan, Pak-Kuen	Delay Models for MOS/LSI Circuits	Architecture	M. Ercegovac	September 1987
Kelem, Steven H.	PEANUTS - A Personalized Easy Access Network Users' Terminal System	Architecture	M. Ercegovac B. Bussell	September 1987
Lindell, Steven	Query Theory and the Complexity of Parallel Computations	Theory	S. Greibach	September 1987
Mueller, Erik Thomas	Machine Daydreaming: A Computational Theory of Emotions and Creative Planning	Artificial Intelligence	M. Dyer	March 1987
Reiher, Peter Lawrence	Naming Issues in Large Scale Distributed Systems	Distributed Systems (Ad Hoc)	G. Popek	September 1987
Takata, Melvin	Interval-Based Timing Simulation Using A Graph Model of Timing Behavior	Architecture	M. Ercegovac	December 1987
Tso, Kam Sing	Recovery In Multi-Version Software	Architecture	A. Avizienis	March 1987
Zernik, Uri	A Process Model for Second Language Acquisition	Programming Languages	M. Dyer	June 1987

TABLE II
STUDENTS WHO ARE IN AN ADVANCED STAGE OF PROGRESS FOR THE PH.D. DEGREE OR HAD THE DEGREE AWARDED DURING 1988(*)

NAME	DISSERTATION TITLE	MAJOR FIELD	ADVISER	EXPECTED COMPLETION DATE
Alkalaj, Leon	An Architectural Model for a Flat Concurrent Prolog Processor	Computer Architecture	M. Ercegovac	June 1989
Alvarado, Sergio	Understanding Editorial Text: A Computer Model of Reasoning and Argument Comprehension	Computer Methodology: Machine Intelligence	M. Dyer	December 1988
Arbab, Bijan	A Formal Language for Representation and Reasoning About Indirect Context	Programming Languages	D.S. Parker	June 1988
Betser, Joseph	Performance Evaluation and Prediction for Large Heterogenous Distributed Systems	Distributed Processing Systems & Architecture	J. Carlyle W. Karplus	December 1988
Chau, Savio	Self-Exercising in Self-Checking Fault Tolerant Computers	Architecture	D. Rennels	December 1988
Chun, Robert	The Fault-Tolerance Characteristics of Neural Networks	Architecture	L. McNamee	June 1989
Dolan, Charles	The Use and Acquisition of Thematic Knowledge	Methodology, Mach. Intelligence	M. Dyer	June 1988
Feng, Da-gan	Practical System Identification Via Compartmental Model Decomposition Techniques	Biocybernetics	J. DiStefano	June 1988
Fleisch, Brett	An Evaluation of Communication Architectures for Distributed Computing Environments	Distributed Operating Systems (Ad Hoc)	G. Popek	December 1988
Goldberg, Arthur	Asynchronous Fault Tolerant Distributed Computing	Network Modeling and Analysis	D. Jefferson	September 1988
Green, Joseph	Load Balancing Algorithms in Computer Networks	Modeling and Analysis	L. Kleinrock	June 1988

Horowitz, Jeffrey	Updating in a Heterogeneous/Distributed Database Environment	Programming Languages and Systems (Data Base Mgt.)	A. Cardenas	June 1989
Huang, Jau-Hsiung	On the Behavior of Algorithms in a Multiprocessing Environment	Modeling and Analysis	L. Kleinrock	June 1988
Joseph, Mark	Architectural Issues in Fault-Tolerance, Secure Computing Systems	Computer System Architecture	A. Avizienis	September 1988
Korfhage, Willard	Distributed Systems with Transient Processors	Modeling and Analysis	L. Kleinrock	December 1988
Landis, Dorothy	CADIS: A Kernel Approach toward the Development of Intelligent Data Management Support for Computer	Architecture	B. Bussell G. Estrin	March 1988*
Lecot, Koenraad	Inexact Reasoning in Expert Systems by Constraint Propagation	Computer Methodology: AI	D.S. Parker	June 1988
Lee, Rei-chi	Query Processing With Domain Semantics	Architecture	W. Chu	December 1988
Lessard, Arthur	Wireless Methods for Factory Automation Environment	Computer Network Modeling	M. Gerla	June 1988
Lichota, Randall	Evaluating Hardware Architectures for Real-Time Parallel Algorithms Using Temporal Specifications	Computer System Architecture	D. Rennels	June 1988
Lor, Kar-Wing	An Assistant for Requirement Driven System Design	Programming Languages & Systems	D. Berry	March 1988*
Lyu, Rung-Tsong	A Design Paradigm for Multi-Version Software	Computer Architecture	A. Avizienis	June 1988
Mehovic, Farid	Performance Analysis of Concurrency Control in Databases	Networks	L. Kleinrock	June 1989
Moore, Donald	Neuromimetic Processing Architectures	Methodology: Machine Intelligence	J. Vidal	December 1988

Moore, Johanna	Enhanced Explanation in Expert and Advice-Giving Systems	Methodology: Physical Sys.	G. Estrin M. Flowers	September 1988
Moreno, Jaime	Design of Special Purpose Arrays for Matrix Computation	Architecture	T. Lang	December 1988
Narain, Sanjai	Log (F) A New Combination of Logic Programming, Rewrite Rules & Lazy Evaluation	Theory	D. Parker	June 1988
Patel, Dorab	Applicative Languages in the Specification and Implentation of VLSI-Oriented algorithms.	System Architecture	M. Ercegovac	December 1988
Paul, Jody	Human Computer Knowledge-Transfer: Facilitating the Communication of Conceptual Knowledge	Machine Intelligence	A. Klinger	March 1988*
Pazzani, Michael	Generalization of Explanatory Schemata	Artificial Intelligence	M. Dyer	June 1988
Schaffa, Frank	Communication Issues on VLSI	Architecture	M. Gerla	September 1988
Schmidt, Phillip	Intelligent Survivability Assessment for Distributed Processing Environments	Modeling & Analysis	M. Gerla	December 1988
Sinai, Bahram	A Systolic Array Architecture for the DTW Pattern Matching Algorithms	Architecture	M. Ercegovac	December 1988
Sit, Chi-Man	Response Time in Distributed Real-Time Systems	Modeling & Analysis	M. Gerla	June 1989
Smallberg, David	Algebraic Specification and Verification of Protocols Using Temporal Logic	Programming Languages & Systems	D. Berry	June 1989
Spetka, Scott	Distributed Operating System Support for Distributed Data Intensive Applications in a Local Area Network Transparent Environment	Data Management	G. Popek	June 1989

Sun, Yea-Li	Topologies and Protocols for LAN Interconnection	Network Modeling & Analysis	M. Gerla	September 1988
Warrier, Unnikrishnan	On The Integration of Petri Net and Queueing Network Theory	Networks Modeling & Analysis	M. Gerla	December 1988
Wu, Kuo-Hsiung	A Design-Conceiving Tool (DCT) Environment for VLSI Building Blocks Designs	Architecture	D. Rennels	December 1988

TABLE III
M.S. DEGREES AWARDED IN CALENDAR YEAR 1987

NAME	TITLE	CHAIR (T) THESIS OR (C) COMPREHENSIVE EXAMINATION	COMPLETION DATE
Aguilera, Christina	Findphrases: A Software Tool	D. Berry (T)	December 1987
Aiken, Timothy A.	A Denotational Semantics Prototyping System	D. Martin (T)	September 1987
Boehnlein, Robert C.	An Object Oriented Data Model for Representing Images	A. Klinger (C)	December 1987
Boz, Mikhail A.	Fault-Tolerant Multi-Processing System	W. Chu (T)	June 1987
Chen, Jia-Hong	Title Not Available	A. Avizienis (C)	December 1987
Chen, Li-Whei	An Expert System for Determining the Dimension of a Sum of Exponentials Solution for Compartmental Model	J. DiStefano (T)	September 1987
Choi, Charles H.	Integration of Pictorial Data Types in the Heterogenous Distributed Database	A. Cardenas (T)	December 1987
Chung, Myungsook A.	Title Not Available	D.S. Parker (C)	June 1987
Dianysian, Alex	Acknowledgement Arc Removal in Data Flow Graph	J. Vidal (T)	December 1987
Eickemeyer, John S.	Approaches to Modeling Flame Using Computer Animation	M. Ercegovac (T)	December 1987
Geiger, Dan	Formal Properties of Dependency Models	J. Pearl (T)	December 1987
Guy, Richard G. II	Reconciliation in a Distributed, Replicated Data Storage Environment	G. Popek (T)	June 1987
Horng, Ming-Yun	An Analytical Model for Packet Flow in a Boolean N-Cube Interconnection Network	L. Kleinrock (T)	December 1987
Kung, Fan-Yi	Title Not Available	J. Carlyle (C)	June 1987
Kwan, Sydney Yau-Shing	Title Not Available	M. Ercegovac (C)	March 1987

Mazer, Alan Scott	APL Implementation on a Message-Based Multiprocessor	M. Ercegovac (C)	March 1987
Morris, Michael J.	Title Not Available	R. Muntz (C)	September 1987
Pham, Quoc Tuan	Parallel Connected Component Labeling and Contour Filling for Binary Images	S. Greibach (T)	September 1987
Schneberger, David L.	Experiments in Image Compression	A. Klinger (T)	June 1987
Schuetz, Werner	Diversity in N-Version Software: An Analysis of Six Programs	A. Avizienis (T)	December 1987
Severin, Larry E.	A Prolog Architecture	A. Klinger (T)	March 1987
Seyedzadeh, Allen	Adding Modules to Prolog	D.S. Parker (T)	March 1987
Stern, Richard A.	Title Not Available	T. Gray (C)	June 1987
Sugarman, Mark J.	Simulation and Classification of Probabilistic Automata	J. Carlyle (T)	December 1987
Sundius, David W.	Title Not Available	R. Muntz (C)	June 1987
Swain, Barbara Joan	Coverage Testing of Multiversion Software	A. Avizienis (T)	March 1987
Takata, Kris K.	INDX, An Automated Indexing Program	D. Berry (T)	June 1987
Wilson, James R.	Title Not Available	M. Gerla (C)	September 1987

TABLE IV
STUDENTS WHO ARE IN AN ADVANCED STAGE OF PROGRESS FOR THE M.S. DEGREE OR HAD THE DEGREE AWARDED DURING 1988(*)

NAME	DISSERTATION TITLE	ADVISER	EXPECTED COMPLETION DATE
Baum, Tsvi	Integrated Services Digital Network (ISDI)	M. Melkanoff (T)	June 1988
Chomut, Tuval	Exploratory Data Analysis in Parallel Coordinates	E. Gafni (T)	June 1988
Dai, Han-Sen	Title Not Available	M. Gerla (C)	June 1988
Dhir, Maneesh	A Performance Analysis Tool for The SARA/IDEAS System	G. Estrin (T)	March 1988*
Domae, Terrance	Operating System Bridges	M. Gerla (T)	June 1988
Heisey, Ingeborg	A Connectionist Model for Low Level Color Vision	J. Skrzypek (T)	June 1988
Kobayashi, Sandra	AL: A Component Evaluation Tool	J. Vidal (T)	June 1988
Ku, Chi	Title Not Available	W. Karplus (C)	June 1989
Lee, Betty	Piping Using Primitives	B. Bussell (T)	March 1988*
Lee, Cheng-Chie	Title Not Available	W. Karplus (T)	June 1988
Lin, Chang-Hsin	3-D Silicon Solution to Lightness Constancy	J. Skrzypek (T)	June 1988
Lin, Tzung-I	On the Analysis of State Restoration in a Fault Tolerant Distributed System	L. Kleinrock (T)	June 1989
Margalit, Gil	Title Not Available	E. Gafni (C)	June 1988
Matsumoto, Atsuko	EVE: Enhanced Viewpoint Enviroment	M. Gerla (T)	September 1988
Molini, Stephen	Title Not Available	A. Cardenas (C)	March 1988*
Parker, Earl	Title Not Available	J. Vidal (C)	June 1988
Pemberton, Joseph	Generalized Digital Perceptrons	J. Vidal (T)	December 1988
Schooler, Eve	Distributed Debugging in a Loosely-Coupled System	L. Kleinrock (T)	March 1988*
Shah, Jayesh	Transparent Protocols for Interconnection of Local Area Networks	M. Gerla (T)	March 1988*

Stiber, Michael	A Connectionist Architecture for Visual Pursuit	J. Skrzypek (T)	June 1988
Tai, Tsung-Yuan	Source Routing in Interconnected Local Area Networks	M. Gerla (T)	March 1988*
Vijayakumar, Tella	Analysis of Tree-net Protocols for Fiber Optic Metropolitan Networks	M. Gerla (T)	June 1988
Weinstein, Matthew	Distributed Resource Control	G. Popek (T)	September 1988
Whitney, Richard	Mapping Meaning Representations from Higher Order Logic to First-Order Forms for Natural Language Access to Databases	J. Pearl (T)	June 1988

COMPUTER SCIENCE DEPARTMENT STAFF MEMBERS

The UCLA Computer Science Department is blessed with a very capable and diligent staff, without whom the department could not function. We give them special thanks for jobs well done.

CSD Departmental Administration: **1st Row:** Mary Beth Crain, Judy Williams, Roberta Nelson, Doris Sublette, Verra Morgan, Arlene Weber, Rosemary Murphy. **2nd Row:** Marta Cervantes, Brenda Ramsey, Tenley Boehm, Alexandra Pham, Saba Hunt, Susanna Reyn, Larry Prince. **Absent:** Steve Sakamoto, Mark Larouche, Rich Wales, Nina Roop.

CSD Contract And Laboratory Staff: **1st Row:** Thuvan Nguyen, Cheryl Childress, Valarie Aylett, Bill Davis, Marilyn Kell. **2nd Row:** June Myers, Gina George, Jackie Trang, Lily Chien. **Absent:** Monique Banarrosh.

VISITING SCHOLARS — 1987

NAME	AFFILIATION	SPONSOR
Boris Y. Kogan	USSR	Walter Karplus
Rudolphe C. Michel	Institut National De Recherche En Informatique et en Automatique France	Michel A. Melkanoff
Kazuyoshi Oshima	Mitsubishi Electric Corporation Japan	Mario Gerla
Kazuaki Oya	Engineer Systems Development Japan	David Rennels
Stefano Rigobello	Telettra Italy	Mario Gerla
Hideo Shimazu	NEC Corporation Japan	Michael G. Dyer

THE COMPUTER SCIENCE DEPARTMENT
INDUSTRIAL AFFILIATES PROGRAM

The Computer Science Department values close links with industrial organizations as a crucial ingredient in assuring the continuing vitality of its academic and research activities. The Industrial Affiliates Program constitutes an important vehicle to that end. The current participants in this program are listed on the following page together with the Computer Science Department faculty members who represent the department as a liason to the corresponding companies.

Industrial affiliates are frequently called upon for advice and suggestions regarding technical and curricular decisions facing the Computer Science Department. Graduate students in advanced stages of their M.S. and Ph.D. programs also turn to the industrial affiliates for advice regarding career plans. In past years, representatives from the industrial affiliate organizations have been invited to participate in the annual Computer Science Department Retreat to facilitate the informal exchange of ideas. Members of the Industrial Affiliate Program enjoy preferential access to some of the facilities of the department such as the Computer Science Department Archives (a special library maintained by the department). The Computer Science Department is always interested in enlarging and broadening the membership of the Industrial Affiliates Organization. Interested company representatives are invited to contact:

Professor Walter J. Karplus
Computer Science Department
University of California
3732 Boelter Hall
Los Angeles, CA 90024-1596

(213) 825-2929

1987

INDUSTRIAL AFFILIATES OF THE COMPUTER SCIENCE DEPARTMENT

Bell Communications Research
Livingston, NJ 07039
UCLA Faculty Sponsor - Wesley Chu

Gould Computer Systems, Inc.
Fort Lauderdale, FL 33313
UCLA Faculty Sponsor - Gerald Estrin

Hewlett Packard
Palo Alto, CA 94304
UCLA Faculty Sponsor - Gerald Estrin

Hitachi, Ltd.
Tokyo, Japan
UCLA Faculty Sponsor - Walter Karplus

IBM
Los Angeles, CA 90025
UCLA Faculty Sponsor - Michael Dyer

Mitsubishi
Japan
UCLA Faculty Sponsor - Mario Gerla

NCR Corporation
Dayton, OH 45479
UCLA Faculty Sponsor - Gerald Estrin

Nippon Electric Corp.
Kawasaki, Japan
UCLA CSD Faculty Sponsor - Walter Karplus

Northrop
Los Angeles, CA 90067
UCLA CSD Faculty Sponsor - Michel Melkanoff

Siemens
Munich, Germany
UCLA Faculty Sponsor - Walter Karplus

Telettra
Italy
UCLA Faculty Sponsor - Mario Gerla

UNISYS (SDC)
Santa Monica, CA 90406
UCLA Faculty Sponsors -
Daniel Berry, Richard Korf

TRW Defense & Space System Group
Redondo Beach, CA 90278
UCLA Faculty Sponsor - Walter Karplus

Xerox
El Segundo, Ca 90245
UCLA Faculty Sponsor - Richard Muntz

DEPARTMENT ORGANIZATION

The Computer Science Department is one of six academic departments within the School of Engineering and Applied Science (SEAS). Administrative responsibility for the direction of the Department rests with Gerald Estrin, Chair of the Department. Jack Carlyle is Vice Chair for Planning and Resources, Sheila Greibach is Vice Chair for graduate affairs, and Lawrence McNamee is Vice Chair for undergraduate affairs. Walter J. Karplus and Jack Carlyle are Co-Directors of the Center for Experimental Computer Science. Arlene Weber is the Management Services Officer and has a wide range of responsibilities regarding day to day administrative operation. Most faculty members participate in the administration of the Department by serving on committees and/or assuming responsibility for one or more functions. The key elected and appointed committees are given below and may give a perspective on the degree of self governance as well as the services and activities available to faculty and students. Students serve on most of the committees and there are also elected graduate student general representatives.

COMMITTEES

Academic Policy Commitee -- *Richard Korf (Chair), Masanao Aoki, Rajive Bagrodia, Wesley Chu, Eliezer Gafni, David Rennels, Bill Fornaciari (graduate student).*

The *elected* Academic Policy Committee reviews all new or modified course proposals of the Computer Science Department, has oversight over existing courses, and establishes guidelines for degree programs and examination procedures used within those programs.

Advisory Committee on TA and Reader Allocations -- *David Kay (Chair), Richard Muntz, Jack Carlyle, Rony Ross (graduate student)*

The TA-Reader Committee makes recommendations about TA and reader allocations as well as policy concerning such allocations.

By-Law 55 Committee -- *Eli Gafni (Chair), Milos Ercegovac, Richard Muntz, D. Stott Parker, Judea Pearl, Jacques Vidal*

The *elected* By-Law 55 Committee advises the chair on faculty merit increases and on the appointment of temporary academic personnel.

Computing Advisory Committee -- *Mario Gerla (Chair), David Kay, Rajive Bagrodia, Jaime Moreno (graduate student), Richard Guy (graduate student)*

The Computing Advisory Committee advises the Chair on all aspects of policy and practice affecting the computing resources and their use in support of department teaching and research.

Space Planning Committee -- *Michael Dyer (chair), Jack Carlyle, Leonard Kleinrock, Walter Karplus, Dave Shrader (graduate student), Arlene Weber (ex oficio).*

The School of Engineering and Applied Science is due to acquire a new building in 1989. Some of the departments will move into the new facility; those, such as the Computer Science Department, remaining in Boelter Hall will receive badly needed increments to their space. The Space Planning committee advises the Chair on allocation of the current space resources and coordinates planning for the space to be available following the proposed retrofit of Boelter Hall.

Computer Science Graduate Student Representatives -- *Greg Frazier, Othar Hansson, Mathew Merzbacher, Joseph Pemberton, Curt Powley, Maria Pozzo.*

Beginning in Fall 1985, students were encouraged to sit in on faculty meetings and represent student interests and concerns. These graduate students report what happens at faculty meetings and act as intermediaries between the students, staff, and faculty. In addition to students serving on departmental committees, 6 general representatives are elected.

ADMINISTRATIVE ASSIGNMENTS

In addition to the formal committee assignments listed above, there are many specific administrative tasks to be carried out by faculty and staff. Some of these functions are listed below.

Advisory Admissions Committee -- *Sheila Greibach, Vice Chair and Graduate Advisor, Margot Flowers, Eliezer Gafni, Mario Gerla, Walter Karplus, D. Stott Parker, Yuval Tamir.*

The Vice Chair for Graduate Studies is in charge of reviewing the dossiers of all applicants for admission to graduate studies, considering the award of financial assistance to incoming graduate students, and assigning faculty advisors to new students, with the advice and assistance of the Admissions Committee representing the various fields of the graduate program.

Archives -- *Doris Sublette (Archivist), Sheila Greibach (Faculty Advisor)*

The Computer Science Department Archives are described below under Department Activities.

Center for Experimental Computer Science -- *Jack Carlyle, Walter Karplus, Leonard Kleinrock*

In 1982 the Center for Experimental Computer Science was formed as a unit within the Computer Science Department, to coordinate and integrate the various research laboratories and environments within the department as well as to develop and operate an advanced research network. Walter Karplus and Jack Carlyle are the Co-Directors of the Center.

Colloquia Development -- *Eliezer Gafni, Michael G. Dyer*

Starting in 1982, Thursday (4-6 P.M.) and Tuesday, Thursday (12-2 P.M.) have been reserved for colloquia and department meetings in an attempt to increase faculty/teaching and faculty/graduate student interaction. This committee is responsible for organizing the CSD Distinguished Lecture Series and structuring CS201 presentations.

Computer Services Coordination -- *Leon Levine*

Leon Levine is the Department's interface with the campus-wide Office of Academic Computing, explaining the Department's computing needs for teaching and research, preparing the computer support budget for the department. He is also responsible for the assignment of computer time to the many computer science classes.

Faculty Recruiting -- *Milos Ercegovac, Saba Hunt*

For 1987-1988, the Computer Science Department has an allocation to recruit at least one new ladder faculty member. The recruiting effort is coordinated by Milos Ercegovac with the help of the Area Groups.

Field Committee Chairs -- *Sheila Greibach (Computer Science Theory), Walter Karplus (Scientific Computing), Leonard Kleinrock (Computer Network Modeling and Analysis), D. Stott Parker (Programming languages and Systems), Richard Korf, (Artificial Intelligence) David Rennels (Computer System Architecture),*

The academic program of the Computer Science Department has been divided into six major areas, corresponding to professionally recognized disciplines within computer science. The faculty who teach and direct research in a Field constitute the Field Committee which is responsible for the overall supervision of the academic program of its area and is consulted regarding matters such as faculty recruitment impinging on its program.

Honors and Recognition -- *Wesley Chu, Allen Klinger*

Throughout the year the Department nominates a number of its faculty members and students for various honors and awards. These include recognition of distinguished teaching, research and academic performance. Wesley Chu is in charge of the coordination of the selection of suitable nominees and the preparation of the necessary documentation.

Placement Examination -- *David Kay*

During registration week of every quarter, the Computer Science Department gives a series of placement examinations. Students with substantial knowledge of computer programming may take these examinations in order to be exempted from some or all of the Computer Science

Department lower division requirements: CS 11, 12 and 13. Mr. Kay is charged with the administration of these examinations.

Preliminary Examinations -- *Masanao Aoki, Allen Klinger*

The Computer Science Department administers, twice each academic year the Major Field Examinations which are the preliminary examinations required of all Ph.D. candidates. Masanao Aoki is charged with the coordination of these examinations.

Quarterly -- *Sheila Greibach*

This activity is described later.

Relations with Industry -- *Walter Karplus, June Myers*

There are several important programs with industry and professional organizations which have recently increased in importance. The Computer Science Department Affiliates have grown in number to 13 and include some of the leading industrial organizations in our field.

Research Review -- *Alfonso F. Cardenas (1988), June Myers*

The Research Review Chair is appointed on an annual basis. The Research Review is described later.

Student Affairs Office (Graduate) -- *Sheila Greibach, Vice Chair for Graduate Affairs, Verra Morgan, Student Affairs Officer, Rosemary Murphy, Student Affairs Assistant*

The Graduate Student Affairs Office, which reports to the Vice Chair for Graduate Affairs, deals with student academic affairs and is responsible for the administration of all aspects of the graduate degree programs and processing of all relevant forms, from the selection of applicants for admission to the granting of M.S. and Ph.D. degrees. In addition, this office administers Teaching Assistantships, Fellowships and Graduate Division Travel Grants and serves as a liaison between computer science graduate students and industry with respect to recruitment.

Student Affairs Office (Undergraduate) -- *Lawrence McNamee, Vice Chair for Undergraduate Affairs, Christina Aguilera, Counselor.*

Department Administrative Office -- *Arlene Weber, MSO, Brenda Ramsey, Purchasing & Receiving, Roberta Nelson, General Personnel Assistant, Ann Goodwin, Accountant, Marta Cervantes, Programmer, Judy Williams, Academic Personnel Assistant.*

Student Organizations -- *David Kay*

Various student organizations housed in the Department are described in detail later. David Kay has overall cognizance of these activities.

Suite Supervisors -- *David Kay, Joseph DiStefano, Walter Karplus*

The faculty of the Computer Science Department is mostly housed in five office suites, located on the third and fourth floors of Boelter Hall. One faculty member in each pair of these suites is responsible for the supervision of secretarial service. Daniel Berry is responsible for suites 3531/3532, Walter Karplus for suites 3731/3732 and Joseph DiStefano for suite 4731.

SEAS AND INTERSCHOOL COMMITTEES

SEAS Executive Committee -- *Sheila Greibach*

SEAS Legislative Assembly -- *Masanao Aoki, Algirdas Avizienis, Alfonso F. Cardenas, Eli Gafni, Lawrence McNamee, Richard Korf*

MCS/MBA Joint Program Development -- *Alfonso Cardenas, Michel Melkanoff, Ephraim McLean*

A joint program with the School of Management leading to a double Master's degree: an MBA and a M.S. in Computer Science.

DEPARTMENTAL ACTIVITIES

In addition to its regular academic and research activities, the Computer Science Department supports a variety of activities designed to increase the effectiveness of its programs and the well being and morale of its faculty and student body. Some of these are described below.

Archives

The Computer Science Department maintains a large and growing collection of books, journals and technical reports relevant to the academic and research programs in its fields. Some of the material in the Archives is donated by faculty members, friends, industrial affiliates or is acquired by various sponsored research projects. In 1979, through the generosity of her colleagues, family, and friends, the **Nancy Lord Ross Memorial Collection** was established in the UCLA Computer Science Department Archives. The Archives are located in Room 3440 Boelter Hall and are open to all graduate students, staff and faculty of the Computer Science Department.

Quarterly

In January 1973, the Computer Science Department initiated the publication of the *Computer Science Department Quarterly.* The Quarterly presents information regarding faculty resources, academic programs, research activities and administrative matters and short echnical contributions from faculty and student, including tutorial articles and reports on recent research. The Quarterly is made available at no cost to departmental graduate students. A considerable number of academic institutions and researchers throughout the world subscribe to the Quarterly.

Research Review

In 1985-1987, the Computer Science Department joined the Electrical Engineering Department in presenting a two-day Annual Research Review. In 1988, the Electrical Engineering Department was unable to participate and a Computer Science Department Review took place on January 28-29. Approximately 35 members of the Computer Science Department (both faculty and students) discussed the work of their research projects before an audience of about 100 participants from UCLA, other universities, government agencies and industry. An important purpose of the Review is to maintain links with industrial organizations.

Spring Retreat

During spring the faculty of the Department usually participates in a retreat away from campus, often joined by selected graduate students, and sometimes also by a few professors from other campuses of the University of California and representatives of industrial organizations. Major modifications in the Departmental academic and research programs are considered at this meeting rather than at the regular Departmental faculty meetings, which are held on campus. In 1980 and 1981 the retreat was held at the USC Idyllwild Conference Center. In 1982 and 1983 the retreat was held at the Aliso Creek Inn in South Laguna. The 1984 retreat was held the UCLA Lake Arrowhead Conference Center. In 1985 and 1986 retreats were held at the UCLA Malibu Conference Center.

Seminars

On one afternoon each week throughout the school year, the Department sponsors a seminar designed to provide its graduate students with a perspective of current work and activities in the computer science field. The speakers include researchers at other universities and in industry as well as from our own faculty. Additional seminars are also held at other times during the academic year. Following is a list of seminars given by the Computer Science Department Lecture Series Program.

FALL 1987 - WINTER 1988
COMPUTER SCIENCE DEPARTMENT
LECTURE SERIES

9/29/87	Robert Felderman UCLA Computer Science Dept.	The Benevolent Bandit Laboratory: A Testbed for Distributed Algorithms Or "Crime Does Pay"
10/6/87	Dagan Feng UCLA Computer Science Dept.	Graph Theoretic (Cut-Set) Analysis of Compartmental Models With Applications to Biological Systems
10/8/87	Professor Richard Korf UCLA Computer Science Dept.	The Artificial Intelligence Major and Minor Fields
10/13/87	Alan Kay Apple Computer	From Actions To Agents
10/15/87	Professor Uzi Vishkin Computer Science Dept. Tel Aviv University, Israel	Logarithmic Time Optimal Parallel Algorithms for List, Tree and Graph Problems
10/22/1987	Professor Arthur M. Geoffrion UCLA Graduate School of Management	Structured Modeling
10/29/87	Professor Kent F. Smith Computer Science Department University of Utah	Design of Integrated Circuits Using Structured Logic
11/03/87	Professor Mart Molle Computer Science Department University of Toronto	An Old Problem with a New Twist: The Helical Window Token Ring
11/10/87	Dr. David Peleg Computer Science Department Stanford University	A Tradeoff Between Space and Efficiency for Routing Tables
11/17/87	Professor Rajive Bagrodia UCLA Computer Science Dept.	Integrated Design and Performance Evaluation of Distributed Systems
12/01/87	Professor Alfred Inselberg UCLA Computer Science Dept.	Visual Multi-Dimensional Geometry With Applications
01/05/88	Nachum Shacham SRI International	Packet Resequencing in Reliable Data Transfer Protocols
1/12/88	Professor Rajive Bagrodia UCLA Computer Science Dept	Integrated Design and Performance Evaluation of Distributed Systems
1/19/88	Dr. Michael M. Krieger	Law, Computers and Moment Problems

1/26/88	Stephen S. Lavenberg T.J. Watson Research Center	Performance Analysis Of Multisystem Transaction Processing
1/27/88	Randolph Nelson	Delay Cost Scheduling for Timesharing Systems
2/2/87	B. Lubachevsky AT&T Bell Labs	An Efficient Distributed Algorithm for Discrete Even Simulation
2/9/88	Professor John H. Holland Department of Computer & Communication Sciences University of Michigan	Learning in Natural and Artificial Systems
2/16/88	Hagit Attiya Tel-Aviv University, Israel	Reliable Coordination in Asynchronous Systems
2/19/88	Amotz Bar-Noy and Danny Dolev	Families of Consensus Algorithms
3/1/88	Uri Zernik GE Corporate Research & Development	The Visual Lexicon
3/8/88	Maria Klawe IBM Almaden Research Center	Application of Matrix Searching to Dynamic Programming
3/10/88	Nicholas Pippenger IBM Almaden Research Center	Error Correction By Majority Voting
3/31/88	Sakti P. Ghosh IBM Almaden Research Center	Statistical Relational Algebra

STUDENT ORGANIZATIONS

CSUA

The Computer Science Undergraduate Association (CSUA) is the representative body for all Computer Science and Engineering Undergraduate Students. They promote the interests of the students, make scholarly contribution to the department and school, and promote student-faculty interaction. Through the CSUA, undergraduates are given a voice in many departmental affairs including the representation of undergraduate concerns at faculty meetings, facilities meetings, and academic policy reviews.

The CSUA sponsors a Distinguished Lecturer Series, the Consumer Electronics Expo, and an Envening with the Professors. They also provide tutorial references, publish a resume book of CS&E students, print a newsletter, and sponsor various social events. Officers for the current academic year are:

Presisdent:	Mike Best
Vice President:	Allen Demson
Treasurer:	Larry Hurvitz
Secretary:	Carol Yonemori
ACM Representative	Greg Oliveau
Faculty Advisor:	Larry McNamee

ACM Student Chapter

The UCLA Student Chapter of the Association for Computing Machinery represents the ACM by offering the benefits and privileges of professional society membership to students in the computer science community. As ACM members, students receive monthly publications featuring current computer science research and are introduced to ACM's Special Interest Groups that sponsor publications, conferences, and local activities related to their "special interest" in computer science. There are over thirty SIGs including SIGACT(Theory), SIGART (Artificial Intelligence), SIGGRAPH (Computer Graphics), SIGPLAN (Programming Languages), SIGSOFT (Software Engineering), and SIGCAS (Computers and Society). On campus, the UCLA Student Chapter represents the interests of students in the Computer Science Department, the Engineering Society, and the School of Engineering. The Chapter assists in the interchange of information in the computer sciences by inviting speakers from academia and industry and provides a forum for discussion of issues pertinent to computing in general. The UCLA Student Chapter annually hosts its Evening with the Professors, where students and professors are invited to the home of a faculty member to socialize in a warm and congenial atmosphere. It also teaches classes about PC-DOS to students using SEASnet, a network of workstations (PC/AT's) tied into a network of mainframe server machines. These DOS classes give students an introduction to SEASnet and show how to use the workstations in their computer-integrated classes on SEASnet. Another activity is the UCLA Microcomputer Faire which is held in conjunction with Engineers' Week and is hosted by both the UCLA Student Chapters of the ACM and the IEEE. The Faire is a showcase for new microcomputer systems and introduces a campus-wide audience to the myriad of products and services available in the field of microcomputers and consumer electronics. Another activity of the Chapter is participating in the ACM Southern California Regional Programming Contest in which colleges and universities from the entire ACM Southern California region are invited to participate. UCLA has taken part in the contest since 1977, and performed remarkably well by consistently placing first or second. With this consistency, the UCLA teams have advanced to the Nationals and have finished second in 1983 and 1985. The cost for student membership in the national organization is $35.00. For application forms and information please stop by the Engineering Society Lounge, located in 4801 Boelter Hall or call (213) 825-7597.

Honorary Society

In 1974 the students of the Computer Science Department formed a chapter of Upsilon Pi Epsilon, the National Honorary Society in the Computer Science Field. Election to this society constitutes a recognition of superior academic performance and brings with it the obligation to provide some tutorial help to students with academic difficulties.

Computer Club

Since its formation in 1958, the UCLA Computer Club has provided an informal atmosphere for people interested in any aspect of computing to meet others with similar interests and to discuss and occasionally act upon their ideas. Four times a year, the club offers free, non-credit classes in programming languages such as Pascal, Fortran, C, ADA, and COBOL. Classes are also offered in UNIX, statistical packages, and personal computers, on occasion. Time is provided on OAC's IBM 370/3033 for students in these classes. For the benefit of students in club classes, and for students in regular university programming classes as well as OAC users, the club offers free consulting services, and sells floppy disks. The Computer Science Department allows students in the club's UNIX class access to the Locus system running on a variety of VAX and IBM 4300 machines. Membership is open to all University of California regular students, staff, and faculty on a quarterly basis for a $2.00 fee. Association with the Computer Club benefits those who receive its services, and perhaps more intensely, those who attempt to provide them.

APPLICATIONS FOR ADMISSION TO GRADUATE STUDY

Because of budgetary limitations and because of its large current student enrollment, the Computer Science Department has been admitting far fewer students to its M.S. and Ph.D. programs than in previous years. Prospective students are urged to submit their applications for admission as promptly as possible. The UCLA Computer Science Department requires that the Graduate Record Examination (GRE) Aptitude test be taken by all applicants. In addition the UCLA Computer Science Department requires the GRE Advanced Mathematics OR Advanced Computer Science Test. Admission applications not accompanied by GRE scores will not be fully processed until those scores are submitted.

The Computer Science Department processes applications for Fall quarters ONLY. There are NO admissions for Winter and Spring quarters.

A very limited number of non-resident tuition waivers is given by the Graduate Council of the Academic Senate in response to recommendations by departments.

Graduate students registered in the Computer Science Department are required to pursue specific degree objectives by enrolling in one of the formal programs available to them. The following is the current choice of academic programs.

Master of Science in Computer Science
Doctor of Philosophy in Computer Science

The detailed requirements of each of these programs are described in each Spring issue of the *Computer Science Department Quarterly*.

AFFIRMATIVE ACTION STATEMENT

The University of California has embarked upon a vigorous program to remedy imbalances in the racial and sexual composition of its student body and staff. The UCLA Computer Science Department is strongly committed to supporting this policy by making every effort to attract qualified members of minorities and women to its graduate student body, faculty, administrative staff and research staff. The Department endeavors to publicize widely all openings for employment so as to bring them to the attention of interested members of minorities and women. In addition, potential students and employees falling into these categories are given special counseling and advice so as to facilitate their entrance into the University community. Interested members of minority groups and women are invited to contact the Chair or the Management Services Officer of the Computer Science Department for further information.

The Computer Science Department of UCLA is currently seeking qualified candidates for the following academic positions.

Research Assistantships

Bachelor of Science in Engineering or Computer Science. Graduate student status at UCLA. Several positions open.

Teaching Assistantships

Admitted into the graduate degree program in order to be eligible for appointment for the next academic year. Computer experience required.

Research Staff Positions

The Center For Experimental Science has research staff positions as Computer System Designer, Principal Programmer and Programmer.

Faculty Positions

Ph.D. in Computer Science and college level teaching experience. To teach and do research in specialized areas of computer science. In particular our areas of primary interest are: Artificial Intelligence, Computer Science Theory, Scientific Computing, Computer Network Modeling and Analysis, Computer Systems Architecture, Programming Languages and Systems. Most new faculty members begin their academic careers at the Assistant Professor level. Tenured positions of Associate Professor or Full Professor are only offered under very unusual circumstances to a candidate with exceptional research accomplishments. In-Residence positions at all levels have the same duties and privileges as the unmodified title and must satisfy the same evaluation criteria, the only difference being that there is no long term commitment of state funds to the position.

Adjunct Faculty Positions

Distinguished accomplishments in Computer Science or related disciplines. This position does not normally carry a salary except for stipends for teaching regularly scheduled courses and/or participation in extramurally sponsored research programs.

Visiting Professorships, Visiting Lecturers, Post Doctoral Scholars, Research Engineers

Individuals possessing a Doctorate or equivalent experience in Computer Science or related disciplines. Candidate may be on leave from an academic position in a recognized university.

TOP-DOWN FUNCTIONAL DECOMPOSITION IN DIGITAL PERCEPTRONS
Gabriela Adler & Jacques J. Vidal
CSD-870014 $4.25

This report examines a radical approach to the parallel processing of Boolean functions, namely the use of large combinatorial networks where the nodes are individually programmable logic modules. The report describes a general top-down approach for programming the nodes and show that the resulting procedures are amenable to concurrent execution.

IN SEARCH OF EFFECTIVE DIVERSITY: A SIX-LANGUAGE STUDY OF FAULT TOLERANT FLIGHT CONTROL SOFTWARE
Algirdas Avizienis, Michael R. Lyu
CSD-870060 $6.75

This report presents the first-year summary of an investigation of fault-tolerant N-version software for automatic flight control that is in progress at the UCLA Dependable Computing and Fault-Tolerant Systems Laboratory. The goals of this research are:

(1) to refine the methods for generating demonstrably effective N-version software in an industrial environment;

(2) to study the extent of effective diversity (dissimilarity) that can be attained by the use of a rigorous methodology (a *design paradigm*) and the choice of different high-level programming languages;

(3) to investigate the extent of correlation of software faults that are uncovered during the development of N-version software, as well as those that are discovered after acceptance;

(4) to estimate the effectiveness and to evaluate the relative safety of N-version software versus single-version software.

This research is directed by Professor Algirdas Avižienis as Principal Investigator. The research team consists of the following graduate students in the the UCLA Computer Science Department: Michael R. Lyu, coordinator, Werner Schütz, Johnny J. Chen, and Chi S. Wu.

Support for the first year of this research has been provided by the Sperry Commercial Flight Systems Division of Honeywell, Inc., Phoenix, AZ, and by the Microelectronics Innovation and Computer Research Opportunities (MICRO) program of the State of California.

Mr. John F. Williams from Honeywell, Inc. has served as the Honeywell technical liaison person and a consultant on flight control computing to this research group. Problem specifications, software design and test procedures, an aircraft model, sets of test cases, and expert advice has been contributed by Mr. Williams and other members of the Honeywell technical staff.

The UCLA research group that is conducting this investigation is fully responsible for the conclusions of this report and for any inaccuracies that may exist.

DEDIX 87 - A SUPERVISORY SYSTEM FOR DESIGN DIVERSITY EXPERIMENTS AT UCLA
Algirdas Avizienis, Michael Lyu, Werner Schutz, Kam-Sing Tso, Udo Voges
CSD-870029 $9.25

To establish a long-term research facility for further experimental investigations of design

diversity as a means of achieving fault-tolerant systems, the DEDIX (DEsign DIversity eXperiment) system, a distributed supervisor and testbed for multi-version software, was designed and implemented by researchers at the UCLA Dependable Computing and Fault-Tolerant Systems Laboratory. DEDIX is available on the Olympus local network, which utilizes the Locus distributed operating system to operate a set of several VAX 11/750 computers at the UCLA Center for Experimental Computer Science. DEDIX is portable to any machine which runs on a Unix operating system. The DEDIX system is described and its applications are discussed in this paper. A review of current research is also presented.

THE EVOLUTION OF FAULT TOLERANT COMPUTING AT THE JET PROPULSION LABORATORY AND AT UCLA: 1960-1986
Algirdas Avizienis and David Rennels
CSD-870022 $5.75

This report presents an overview of the origins and evolution of fault-tolerant computing during the years 1960-1986 at the Jet Propulsion Laboratory, Pasadena, California, and at the University of California, Los Angeles, California, U.S.A. A preliminary version of this report was presented at the Symposium on the Evolution of Fault-Tolerant Computing that was organized by IFIP Working Group 10.4 "Reliable Computing and Fault Tolerance" and took place on June 30, 1986 in Baden, Austria.

The entire text (except Appendix C) will appear as a chapter in the book "The Evolution of Fault-Tolerant Computing" that will be published by Springer-Verlag, Vienna, Austria in 1987. Appendix C consists of the paper "Design of Fault-Tolerant Computers" (reference [Aviz 67a]) that was presented at the 1967 Fall Joint Computer Conference in Anaheim, California. To the extent of our knowledge, this paper introduced the term "fault-tolerant computer" and the concept of fault tolerance into technical literature.

PROCESS SYNCHRINIZATION: DESIGN AND PERFORMANCE EVALUATION OF DISTRIBUTED ALGORITHMS
Rajive Bagrodia
CSD-870062 $1.75

The concept of n-party rendezvous has been proposed to implement synchronous communication among an arbitrary number of concurrent, asynchronous processes. The synchronization and exclusion problems associated with implementing n-party rendezvous are expressed succintly in the context of the committee coordination problem. This paper presents a simple solution for the problem and shows how it can be implemented in a variety of ways. The paper also compares the performance of the implementations suggested in this paper with other algorithms for this problem.

ANALYSIS OF THE NUMBER OF OCCUPIED PROCESSORS IN A MULTIPROCESSING SYSTEM
Abdelfettah Belghith
CSD-870003 $12.25

We view a multiprocessor system as a set of P cooperating processors, and a computer job as a set of tasks partially ordered by some precedence relationships, and represented by a directed acyclic graph called a Process Graph. Nodes in the process graph represent the tasks and edges represent the precedence relationships between these tasks.

Many parameters are in play to characterize the underlined multiprocessor system. These are: the job arrival process, the process graph description (number of nodes, number of levels, distribution of the tasks among the levels, and the precedence relationships among the tasks), task processing requirements, and the number of processors in the system.

In this report, we investigate the probability distribution of the number of occupied processors, the generating function of this distribution and its first two moments. In particular, the expected number of busy processors is found to be dependent only on the average number of tasks per job, the job average arrival rate, and the task average processing requirement.

TAXONOMY, STRUCTURE ND IMPLEMENTATION OF EVIDENTIAL REASONING MODELS

Moshe Ben Bassat

CSD-870005 $3.75

(This work was supported in part by the National Science Foundation, Grant DSR 83-13875; Presented at the 2nd AAAI Workshop on Uncertainty in AI, Philadelphia, PA., August 1986.)

Keywords: evidential reasoning, Bayesian inference networks, expert systems, situation assessment

The fundamental elements of evidential reasoning problems are described, followed by a discussion of the structure of various types of problems. Bayesian inference networks and state space formalism are used as the tools for problem representation. A human-oriented decision-making cycle for solving evidential reasoning problems is described and illustrated for a military situation assessment problem. The implementation of this cycle may serve as the basis for an expert system shell for evidential reasoning: i.e., a situation assessment processor.

AN OBJECT ORIENTED METHODOLOGY FOR THE SPECIFICATION OF MARKOV MODELS

Steven Berson, Edmundo Silva, Richard Muntz

CSD-870030 $5.50

Modelers wish to specify their models in a symbolic, high level language while analytic techniques require a low level, numerical representation. The translation between these description levels is a major problem. We describe a simple, but surprisingly powerful approach to specifying system level models based on an object oriented paradigm. This basic approach will be shown to have significant advantages in that it provides the basis for modular, extensible modeling tools. With this methodology, modeling tools can be quickly and easily tailored to particular application domains. An implementation in Prolog, of a system based on this methodology and some example applications are given.

THEORY-W SOFTWARE PROJECT MANAGEMENT: A CASE STUDY

Barry Boehm and Rony Ross

CSD-870044 $6.00

The search for a single unifying principle to guide software project management has been relatively unrewarding to date. Most candidate principles are either insufficiently general to apply in many situations, or so general that they provide no useful specific guidance.

This paper presents a candidate unifying principle which appears to do somewhat better. Reflecting various alphabetical management theories (X, Y, Z), it is called the Theory W approach to software management.

Theory W: Make Everyone a Winner

The paper explains the Theory W principle and its two subsidiary principles: *Plan the flight and fly the plan;* and *Identify and manage your risks.*

To test the practicability of Theory W, a case study is presented and analyzed: the attempt to introduce new information systems to a large industrial cooperation in an emerging nation. The case may seem unique, yet it is typical. The analysis shows that Theory W and its subsidiary principles do an effective job both in explaining why the project encountered problems, and in prescribing ways in which the problems, and in prescribing ways in which the problems could have been avoided.

AN OBJECT-ORIENTED DATA MODEL FOR MANAGING COMPUTER-AIDED DESIGN AND COMPUTER-AIDED MANUFACTURING DATA BASES

Stephanie Jo Cammarata

CSD-870026 $33.00

As a result of strong and steady CAD/CAM (Computer-Aided Design/Computer-Aided Manufacturing) growth over the past 20 years, special facilities for managing design and manufacturing data have been required. CAD/CAM Data Base Management Systems

(DBMS) fill this role. The most widely used CAD/CAM DBMS manage data for only a single CAD or CAM application and cannot integrate graphical, geometrical, manufacturing, and administrative data. Furthermore, current modeling facilities are inadequate for representing semantic features and constraints captured by an engineering drawing. These limitations cause data flow gaps, inconsistant and redundant data, and unnatural data organization in existing CAD/CAM data bases.

The purpose of this dissertation is to develop sophisticated facilities for managing CAD/CAM data bases. This work focuses on mechanical design, engineering, and manufacturing, specifically *product definition data* generated during initial design phases. Based on a detailed analysis of CAD/CAM data management requirements, and interaction with data management and manufacturing personnel at Lockheed Corporation and Rockwell International, I propose the following goals for integrated CAD/CAM DBMS:

* conceptual centralization
* part-oriented BOM hierarchies
* customized representation of assemblies and parts
* incorporation of domain knowledge

The product of this research is the theoretical design of an object-oriented data model, ODM, and the implementation of an ODM computer software prototype supportng CAD/CAM DBMS goals. The ODM software system is written in T, a lexically scoped dialect of Lisp, and currently runs on Vax and Apollo networks in UCLA's Computer Science Department. The ODM system provides the following unique features:

* object-oriented semantic modeling facilities
* dynamic schema capabilities
* semantic contraint maintenance
* heterogeneous data types

I conclude with an evaluation of ODM toward achieving the goals of integrated CAD/CAM DBMS. Data bases supporting Hughes' PWA (Printed Wiring Assembly) and *Producibility Feedback* applications were obtained for evaluation testing. Although most discussion concentrates on mechanical manufacturing; the developed methodology and tools for CAD/CAM data management also apply to other design and manufacturing domains such as architecture and electronics.

BOUNDS ON INVERSE OF THE NODE-CONDUCTANCE MATRIX
Pak K. Chan
CSD-870021 $4.25

In [1] and [2], we have shown that the delay estimation problem can be solved by finding the inverse of the node-conductance matrix GG of a resistive network. The problem of finding $RR \equiv GG^{-1}$ is approached in this report in several ways. First, the use of an exact expression is examined. This method involves examining the geometry of the network and then applying a topological formula to find the desired driving-point resistance. Second, bounds for RR are presented; we attempt to find two matrices $\overline{RR}=(\overline{R}_{i,j})$ and $\underline{RR}=(\underline{R}_{i,j})$ such that $\overline{R}_{i,j} \geq R_{i,j} \geq \underline{R}_{i,j}$, for all i and j. Evaluating these bounds largely involves finding a least-resistance path between the ground node and the rest of the nodes in a network. Third, these bounds can serve as initial guesses in an iterative procedure that would tighten up the delay bounds from both sides.

DISTRIBUTED COMPUTER SIMULATION OF A DATA COMMUNICATION NETWORK
Shun Cheung
CSD-870039 $16.00

Performance evaluation through computer simulation is a very important step in the planning of modern data communication networks. In view of the advent of network of micro-computer workstations, it is attractive to investigate the possibility of carrying out these time and memory-comsuming computations in this type of new environment in a distributed manner.

This dissertation describes the conceptual development and implementation of a distributed discrete-event system simulator for data-communication networks. A slotted-ring network model is used as a benchmark; the model includes multi-terminal (user) nodes and host (cpu) modes. Inherent concurrency and other special features of this type of models have influenced the design of the simulation software, with attention to the selection of appropriate mechanisms for distributed processing, task assignment, synchronization, memory management, congestion control, and data collection. Synchronization is carried out using state-saving and rollback strategies, based on the time-wrap concept, particularized to queueing-network model applications; state information is maintained with the aid of pointer manipulations in a collection of queues. Permits are used to throttle the flow of inter-processor messages in order to prevent congestion in the communication medium and to restrain memory usage in the simulators. A model partitioning algorithm has been developed for ring type network models.

To demonstrate feasibility and correctness of the methodology, a distributed simulator is implemented on an existing minicomputer network supporting distributed processing, with a view toward the longer-range goal of developing tools usable in computing environments consisting of distributed workstations. The experimental results suggest that medium-scale distributed simulation using a network of mini-computers or workstations is feasible and is a promising approach especially for large models which exceed the capacity of single machines.

DISTRIBUTED DATA BASE MANAGEMENT FOR REALTIME BMD APPLICATIONS - FINAL REPORT
Wesley Chu, et. al.
CSD-870059 $16.25

During the past year, we have concentrated our efforts in the following areas of distributed systems: module assignment and scheduling for distributed systems, testbed validation of design techniques, performance of concurrency control algorithm for distributed database systems, and query processing with domain semantic.

1.0 Module Assignment for Real-Time Distributed Processing Systems

Module assignment is a key issue that effects system performance in distributed systems. Response time is intimately related with module assignment. Therefore, we shall use response time as a performance measure in our research. We have investigated two related areas. The first area considers the module precedence effect on module assignment, and the second considers the replicated module assignment to provide load balance and improves response time.

1.1 Module Assignment and Precedence Relations for Distributed Real-Time Distributed Systems

It is well known that module assignment should consider module precedence relationships. However, most of the published task allocation work has not considered the precedence effect. This motivates us to study and understand the effects of precedence relationship (PR) among program modules on response time. A new loading function that includes the Intermodule Communication (IMC) and Accumulative Execution Time (AET) of each module is also proposed. Our study reveals that minimizing the most heavily loaded (bottleneck) processor is a good principle for module assignment. Further, the PR module effect can also be integrated into the above assignment principle. When module PR is considered in the task assignment, it yields better performance than without considering the PR effect. Detailed results are summarized in Chapter 2.

1.2 Module Replication and Assignment for Real-Time Distributed Systems

An analytical model is developed to estimate the task response time of distributed systems. The model considers such factors as interprocessor communications, module precedence relationship, module scheduling, interconnection network delay, and assignment of modules and files to computers. A heuristic algorithm for module assignment is developed to iteratively search for module assignments which provide shorter task response times. Assigning replicated modules may reduce task response time. Therefore, the algorithm also considers module replications. Using the sum of task

response time and penalty delay for the violations of specified thread response time requirements as the objective function, an "optimal" module multiplicity and module allocation can be determined by the proposed algorithm. The detailed model and algorithm are presented in Chapter 2.

Our study reveals that the task response times for a given module assignment (with replications) generated by the algorithm compare closely with that of the simulation and exhaustive search. A series of experiments is also performed to characterize the behavior of the algorithm.

2.0 A Batch Service Scheduling Algorithm with Time-Out for Real-Time Distributed Processing Systems

A new scheduling algorithm for reducing overhead and thus response time is proposed for distributed processing systems. The algorithm groups several module invocations into a batch and processes them together to reduce certain scheduling overhead. A time-out clock is used to avoid excess delay in forming a batch. The clock is set when the first invocation arrives at the batch queue. The batch is formed when either the number of invocations reaches the prespecified maximum batch size or the time-out period ends. We denote this scheduling algorithm Batch Service with Time- out (BST). An analytical model is developed to estimate response time for this scheduling algorithm. The response time of a module using the BST algorithm depends on the invocation rate, scheduling overhead, execution time, maximum batch size, and time-out period. The assumptions used in the model are validated by simulations.

Comparing performance of a system using BST with that of using first-come- first-served (FCFS) scheduling algorithm, we note that the amount of improvement depends on the ratio of the fixed scheduling overhead to the incremental scheduling overhead. At heavy invocation rates, more batches will be formed when using the BST algorithm, therefore fixed scheduling overhead is reduced and more response time improvement can be achieved (See Chapter 3). As a result of reduction in overhead, the system using BST provides more capacity than that of using FCFS.

3.0 Testbed-Based Validation of Design Techniques

During the past three years, we have been jointly working with Unisys SDC at Huntsville to study and develop design methodology for tightly coupled distributed systems. Experimentation on the testbed provides us with insights on algorithm behavior. We have developed a fault tolerant locking (FTL) algorithm for the tightly coupled multiple processing system [1], designed the experiments, and studied its feasibility and performance. Experimental results reveal that the FTL is capable of detecting a processor failure during update and recovering data inconsistency among replicated copies. The overhead for performing the fault-tolerant locking protocol depends on the lock frequency and its application. The parameters that may affect system performance are: time-out period, lock granularity (record or a group of records), and lock protocol (e.g., exclusive lock for write and shared for read, or reserve, upgrade, or exclusive lock).

We have also used the testbed for studying the performance of lock granularity (e.g. record, file) and the performance of reserve-upgrade locking protocol. Because of the read/write pattern of the radar tracking application, the results reveal that simple record locking provides better response time than file locking and reserve-upgrade locking. The detailed results are summarized in Chapter 4.

4.0 Performance of Concurrency Control Algorithms for Real-Time Distributed Database Systems

The survivability of distributed systems can be improved with multiple copies of files. When an update is performed on a copy, the update should be written on all other file copies. If the computer that is handling the update fails during the update process, all the copies may not be updated, resulting in mutual inconsistency.

To study the different concurrency control techniques, we introduced new performance measures such as accuracy and weak consistency for characterizing the performance for data consistency. We have experimentally studied three types of protocols via simulation: Primary Site Locking (PSL) [2], Exclusive Writer Protocol (EWP) [3], and Time Stamp with

modified Rollback (TMR) [4] and compared their performance in terms of response time, communication overhead, query rate, update/query ratio, consistency, and accuracy. Detailed discussions are given in Chapter 5. Protocols that assure weak consistency yield better performance (response time) than those assuring strong consistency. The data consistency requirement for different types of data is application dependent. Further, it also depends on how the data is used for decision making. It is possible that when data state is x, weak consistency is sufficient. While data state becomes y, strong consistency is required.

5.0 A Methodology of Knowledge Acquisition for Semantic Query Processing

Query processing is a key consideration in database management systems. Conventional approach uses a domain-independent approach for query processing design. Queries are transformed algebraically to determine the optimal access plan for retrieving the answer. *Semantic query optimization* uses a set of integrity constraints and reasoning to transform a given query into a different but more efficient query yet yields the same answer. It has been shown that this technique has great promise for improving system performance. However, knowledge acquisition problem needed to be solved before this technique can be of practical use.

Data modelling provides a useful tool for database design and usage. Conventional data models such as *hierarchical, network,* and *relational* models provide a record-based data structure for modelling database application. However, because of the lack of expressiveness of the conventional data models in modelling various database applications, semantic data models have been developed to provide a rich set of semantic constructs to describe various situations of the application. Database designer then uses his knowledge about the application to define the database schema. Most of the semantic data models focus only on providing structural specification and ignore the importance of knowledge for designing the database. However, this knowledge is very useful to semantic query processing and should be saved and used in query processing.

In this research, a semantic data model is developed that provides a knowledge specification capability associated with the semantic constructs for schema specification. This data model provides not only the necessary semantic expressive capability to model various database applications, but also specifies domain knowledge which can be used to improve query processing performance. A knowledge acquisition tool is developed for systematically collecting useful domain knowledge for semantic query processing. A semantic database management system (SDBMS) is proposed that integrates semantic data modelling with semantic query processing. SDBMS provides facilities to systematically acquire semantic knowledge and use them to improve query processing. Detailed discussions are given in Chapter 6.

THE INDUCTIVE LOGIC OF INFORMATION SYSTEMS
Norman C. Dalkey
CSD-870034 $2.75

(This work was supported in part by National Science Foundation Grant IST 84-05161)

An inductive logic can be formulated in which the elements are not propositions or probability distributions, but information systems. The logic is complete for information systems with binary hypotheses, i.e., it applies to all such systems. It is not complete for information systems with more than two hypotheses, but applies to a subset of such systems. The logic is inductive in that conclusions are more informative than premises. Inferences using the formalism have a strong justification in terms of the expected value of the derived information system.

MODELS VS. INDUCTIVE INFERENCE FOR DEALING WITH PROBABILISTIC KNOWLEDGE†
Norman C. Dalkey
CSD-870050 $1.75

†Department of Computer Science, University of California, Los Angeles, CA. 90024. This work was supported in part by National Science Foundation Grant IST 84-05161.

1. Introduction

Two different approaches to dealing with probabilistic knowledge are examined -- models and inductive inference. Examples of the first are: influence diagrams [1], Bayesian networks [2], log-linear models [3, 4]. Examples of the second are: games-against nature [5, 6], varieties of maximum-entropy methods [7, 8, 9], and the author's min-score induction [10].

In the modeling approach, the basic issue is manageability, with respect to data elicitation and computation. Thus, it is assumed that the pertinent set of users in some sense knows the relevant probabilities, and the problem is to format that knowledge in a way that is convenient to input and store and that allows computation of the answers to current questions in an expeditious fashion.

The basic issue for the inductive approach appears at first sight to be very different. In this approach it is presumed that the relevant probabilities are only partially known, and the problem is to extend that incomplete information in a reasonable way to answer current questions. Clearly, this approach requires that some form of induction be invoked. Of course, manageability is an important additional concern.

Despite their seeming differences, the two approaches have a fair amount in common, especially with respect to the structural framework they employ. Roughly speaking, this framework involves identifying clusters of variables which strongly interact, establishing marginal probability distributions on the clusters, and extending the subdistributions to a more complete distribution, usually via a product formalism. The product extension is justified on the modeling approach in terms of assumed conditional independence; in the inductive approach the product form arises from an inductive rule.

A LOGIC OF INFORMATION SYSTEMS
Norman C. Dalkey
CSD-870057 $2.50

A logic can be formulated with information systems as elements. The calculus of this logic is similar to, but not identical with, Boolean algebra. The logic is inductive--conclusions have more information than premises. Inferences have a strong justification; they are valid for all proper scoring rules.

REMOVING REDUNDANCIES IN CONSTRAINT NETWORKS
Avi Dechter, Rina Dechter
CSD-870006 $3.00

The removal of inconsistencies from the problem's representation, which has been emphasized as a means of improving the performance of backtracking algorithms in solving constraint satisfaction problems, increases the amount of redundancy in the problem. In this paper we argue that some solution methods might actually benefit from using an opposing strategy, namely, the removal of redundancies from the representation. We present various ways in which redundancies may be identified. In particular, we show how the path-consistency method, developed for removing inconsistencies can be reversed for the purpose of identifying redundancies, and discuss the ways in which redundancy removal can be beneficial in solving constraint satisfaction problems.

MINIMAL CONSTRAINT GRAPHS
Avi Dechter, Rina Dechter
CSD-870007 $5.00

(This work was supported in part by the National Science Foundation, Grant #DCR 85-01234, and by the AISAF project sponsored by the U.S. Army's Signals Warfare Center.)

Strong relationships exist between the topological properties of the constraint graph of given constraint satisfaction problem (CSP) and the computational tractability of that problem. Generally speaking, it is best for the constraint graph to have as few edges as possible. This paper deals with two main issues. First, we discuss the type of information exhibited by the constraint graph and the relevance of this information to the solution of the CSP. Second, we consider the possibility of changing the representation of a given

CSP in order to obtain a more informative constraint graph. Observing that ordinary path-consistency operation usually adds edges to the network, we propose a modified path-consistency scheme preventing this from occurring. We then develop two approximation algorithms (which do, in some sense, the reverse of path-consistency) for reducing the connectivity in the constraint graph, and prove their properties.

DECOMPOSING AN N-ARY RELATION INTO A TREE OF BINARY RELATIONS
Rina Dechter
CSD-870011 $1.25

(This work was supported in part by the National Science Foundation, Grant #DCR 85-01234)

We present an efficient algorithm for decomposing an n-ary relation into a tree of binary relations, and provide an efficient test for checking whether or not the tree formed represents the relation. If there exists a tree-decomposition, the algorithm is guaranteed to find one, otherwise, the tree generated will fail the test, then indicating that no tree decomposition exist. The unique features of the algorithm presented in this paper, is that it does not apriori assume any dependencies in the initial relation, rather it derives such dependencies from the bare relation instance.

AN INTEGRATED STRATEGY FOR IMPROVED BACKTRACK
Rina Dechter
CSD-870010 $5.75

Keywords: problem solving, learning, constraint-satisfaction, backtracking, cycle-cutset.

In previous work researchers in the areas of Constraint-Satisfaction Problems (CSPs) and Prolog had suggested various enhancements to the Naive Backtrack algorithm. Each scheme was presented and tested individually, and the average performances were compared.

The contribution of this paper is in devising a backtrack strategy that integrates three improvement schemes: "*Backjump*", "*Learning while searching*" and the *cycle-cutset method*"; Backjump and the cycle-cutset method work best when the constraint-graph is sparse, while the learning scheme mostly benefits problem instances with dense constraint graphs. The *Integrated-strategy* proposed here lets each scheme dominates when instances favorable to its performance are presented and makes them cooperate on intermediate instances. The experiments show that, in hard problems, the average improvement realized by the integrated scheme is by 20-25 % higher then any of the individual schemes.

A CONSTRAINT-NETWORK APPROACH TO TRUTH-MAINTENANCE
Rina Dechter
CSD-870009 $3.25

(This work was supported in part by the National Science Foundation, Grant #DCR 85-01234)

Track: Science track
Topic area: Automated Reasoning.
Keywords: belief revision, constraint-networks, diagnosis, truth-maintenance .

This paper presents a constraint-network formulation for maintaing consistency of beliefs in dynamically changing knowledge bases. It exploits techniques developed for Constraint-Satisfaction problems and provides an efficient distributed scheme for updating beliefs in singly connected constraint-networks.

We present a belief-revision process consisting of two phases. In the first phase, called *support-propagation*, each variable updates the number of extensions consistent with each of its values. The second, called *diagnosis*, is invoked by a variable that detects a contradiction, and identifies a minimal set of assumptions that accounts for the contradiction. The support-propagation phase is accomplished in a single pass through the network while the diagnosis process takes at most 4 passes. Overall, the impact of any new input to the system can be propagated in at most 5 passes through the network. Extensions of this scheme to multiply-connected networks using the cycle-cutset and clustering approaches, are also discussed.

NETWORK-BASED HEURISTICS FOR CONSTRAINT-SATISFACTION PROBLEMS
Rina Dechter and Judea Pearl
CSD-870037 $11.75
(This work was supported in part by the National Science Foundation Grant, DCR 85-01234.)

Many AI tasks can be formulated as Constraint-Satisfaction problems (CSP), i.e., the assignment of values to variables subject to a set of constraints. While some CSPs are hard, those that are easy can often be mapped into sparse networks of constraints which, in the extreme case, are trees. This paper identifies classes of problems that lend themselves to easy solutions, and develops algorithms that solve these problems optimally. The paper then presents a method of generating heuristic advice to guide the order of value assignments based on both the sparseness found in the constraint network and the simplicity of tree-structured CSPs. The advice is generated by simplifying the pending subproblems into trees, counting the number of consistent solutions in each simplified subproblem, and comparing these counts to decide among the choices pending in the original problem.

TREE-CLUSTERING SCHEMES FOR CONSTRAINT-PROCESSING*
Rina Dechter, Judea Pearl
CSD-870065 $4.75

(*This work was supported in part by the National Science Foundation, Grant #DCR 85-01234)

The paper offers a systematic way of regrouping constraints into hierarchical structures capable of supporting information retrieval without backtracking. The method involves the formation and preprocessing of an acyclic database to be amortized and maintained over many problem instances. Once found, the database permits a large variety of queries and local perturbations to be process swiftly, either by sequential backtrack-free procedures, or by distributed constraint-propagation processes.

THE OPTIMALITY OF A* †
Rina Dechter and Judea Pearl
CSD-870049 $6.25

† This work was supported in part by the National Science Foundation, Grant #IRI 85-01234.

This paper examines the computational optimality of A*, in the sense of never expanding a node that could be skipped by some other algorithm having access to the same heuristic information that A* uses. We define a hierarchy of four optimality types, and consider three classes of algorithms and four domains of problem instances relative to which computational performances are appraised. For each class-domain combination, we then identify the strongest type of optimality that exists and the algorithm achieving it. Our main results relate to the class of algorithms which, like A*, return optimal solutions (i.e., admissible) when all cost estimates are optimistic (i.e., $h \leq h^*$). On this class we show that A* is not optimal and that no optimal algorithm exists, but if we confine the performance tests to cases where the estimates are also consistent, then A* is indeed optimal. Additionally, we show that A* is optimal over a subset of the latter class containing all *best-first* algorithms that are guided by path-dependent evaluation functions.

TREE-CLUSTERING SCHEMES FOR CONSTRAINT-PROCESSING*
Rina Dechter and Judea Pearl
CSD-870054 $4.75

*This work was supported in part by the National Science Foundtation, Grant #DCR 85-01234

The paper offers a systematic way of regrouping constraints into hierarchical structures capable of supporting information retrieval without backtracking. The method involves the formation and preprocessing of an acyclic database to be amortized and maintained over many problem instances. Once found, the database permits a large variety of queries and local perturbations to be process swiftly, either by sequential backtrack-free procedures, or by distributed constraint-propagation processes.

SIMPLE RADIX-4 DIVISION WITH DIVISOR SCALING
Miloš D. Ercegovac and Tomas Lang
CSD-870015 $3.00

A radix-4 division algorithm with divisor scaling is proposed. The algorithm uses a recurrence with carry-save addition and combines simple scaling with a quotient-selection function that depends only on the estimate of the partial remainder and is independent of the divisor. The redundant quotient is converted on-the-fly into a conventional representation without carry-propagate addition. The scheme results in a significant speed-up with respect to both the radix-2 and radix-4 without scaling, with about the same hardware.

FAST MULTIPLICATION WITHOUT CARRY-PROPAGATE ADDITION
Milos Ercegovac and Tomas Lang
CSD-870047 $4.25

A common scheme for fast multiplication keeps the partial products in redundant form (carry-save or signed-digit) and converts the result to conventional representation in the last step. This step requires a carry-propagate adder which is comparatively slow and occupies a significant area of the chip in a VLSI implementation. In this paper we report a scheme that does not require this carry-propagate step. The scheme performs the multiplication most-significant bit first and produces a conventional sign-and- magnitude product by means of an on-the-fly conversion. The resulting implementation is fast and regular and is very well suited for VLSI. The scheme is presented for general radix 4 and radix-4 implementations are described. Error analysis of the implementations is performed and different rounding schemes are considered. Three different implementations are presented: one in which all adders are of the signed-digit type, another in which carry-save adders are used in the main multiplication array, and a third that improves the speed by computing odd and even partial products concurrently. Left copy of tech report on your desk for a # and front page. This is the abstract.

REDUNDANT AND ON-LINE CORDIC: APPLICATION TO MATRIX TRIANGULARIZATION AND SVD
Milos Ercegovac and Tomas Lang
CSD-870046 $5.25

Several modifications to the CORDIC method of computing angles and performing rotations are presented: (i) the use of redundant (carry-free) addition instead of a conventional (carry-propagate) one; (ii) a representation of angles in a decomposed form to reduce area and communication bandwidth; (iii) the use of on-line addition (left-to-right, digit-serial addition) to replace shifters by delays; and (iv) the use of on-line multiplication, square root and division to compute scaling factors and perform the scaling operations. The modifications presented improve the speed and the area of CORDIC implementations. The proposed scheme uses efficiently floating-point representations. We discuss the application of the modified CORDIC method to matrix triangularization by Givens' rotations and to the computation of the single value decomposition (SVD).

ON-LINE SCHEMES FOR COMPUTING ROTATION ANGLES FOR SVDS
Milos D. Ercegovac and Tomas Lang
CSD-870043 $5.75

Two floating-point radix-2 schemes using on-line arithmetic for implementing the direct two-angle method for SVDs are presented. The first scheme is an on-line variant of the cosine/sine approach and is the fastest of the schemes considered: it performs the 2x2 SVD step in about $2n$ clock cycles. However, it requires a relatively large number of modules; this number is reduced when some modules are reused, resulting in a time of $3n$ clock cycles. The number of modules of this on-line version is still larger than that of the conventional one, but this is compensated by the smaller number of bit-slices per module and by the digit-serial communication among modules. The corresponding speed-up ratios are of 5 and 3 with respect to a conventional arithmetic implementation. The

second scheme uses an on-line CORDIC approach and performs the 2x2 SVD in about $7n$ clock cycles and is advantageous because it is more time-area efficient. It results in a speed-up of about 2.5 with respect to the conventional CORDIC implementation.

FAST TRIANGULARIZATION BY GIVENS ROTATION USING ON LINE CORDIC
Milos D. Ercegovac and Tomas Lang
CSD-870045 $3.25

A scheme for triangularization of a matrix using redundant and on-line CORDIC modules is proposed. Its implementation is simpler and faster than implementations using conventional CORDIC modules. The proposed scheme has the following features: • the rotation processors uses angles in a decomposed form thus eliminating the angle recurrence and allowing overlap between the angle processor and rotation processors: no ROMs are required; • the (x,y)-recurrences are transformed so that only one variable shifter is required; • the carry-propagate addition is replaced by a redundant addition (cary-save or signed-digit) thus reducing the clock cycle; • the rotation recurrences are unfolded and implemented in on-line manner thus replacing the variable shifter with simple delays; • the scale factor is computed in on-line manner; • the scheme uses efficiently floating-point representations.

AN IMPROVED CONSTRAINT-PROPAGATION ALGORITHM FOR DIAGNOSIS
Hector Geffner & Judea Pearl
CSD-870012 $4.25

(This work was supported in part by the National Science Foundation Grant DSR 83-13875.)

Diagnosing a system requires the identification of a set of components whose abnormal behavior could explain faulty system behavior. Previously, model-based diagnosis schemes have proceeded through a cycle of *assumptions* → *predictions* → *observations* → *assumptions-adjustment*, where the basic assumptions entail the proper functioning of those components whose failure is not established. Here we propose a scheme in which every component's status is treated as a variable; therefore, predictions covering all possible behavior of the system can be generated. Remarkably, the algorithm exhibits a drastic reduction in complexity for a large family of system-models. Additionally, the intermediate computations provide useful guidance for selecting new tests.

The proposed scheme may be considered as either an enhancement of the scheme proposed in [de Kleer, 1986] or an adaptation of the probabilistic propagation scheme proposed in [Pearl, 1986] for the diagnosis of deterministic systems.

ON THE PROBABILISTIC SEMANTICS OF CONNECTIONIST NETWORKS
Hector Geffner and Judea Pearl
CSD-870033 $2.00

(This work was supported in part by the National Science Foundation, Grant #DCR 83-13875.
†To appear in Proceedings of IEEE First International Conference on Neural Networks, June 1987, San Diego, CA)

The goodness/energy paradigm [Hopfield 82] has recently emerged as a useful framework for the construction and analysis of connectionist models. Its lack of a clear semantics however, makes the framework unsuitable as an specification language for the declarative content of those models. This paper establishes a correspondence between connectionist networks and a well known family of probabilistic networks, thus, endowing connectionist models with a well understood probabilistic semantics. Additionally we show how a natural extension of the energy formulation presented in [Hopfield 82] leads to models capable of expressing arbitrary probability distributions.

SOUND DEFEASIBLE INFERENCE
Hector Geffner and Judea Pearl
CSD-870058 $5.50

A new system of defeasible inference is presented having the following features:

1. Spurious extensions are prevented without forcing one to explicitly enumerate exceptions,

2. The system has a sound, clear probabilistic semantics, guaranteeing that the consequences are highly probable whenever the premises are,

3. The system is clean: proofs can be constructed very much like in natural deduction systems in logic.

Additionals implications of the framework proposed are precise, proof theoretic and semantic accounts of defaults, and a formalization of the notion of irrelevance in the context of non-monotonic reasoning.

A SOUND FRAMEWORK FOR REASONING WITH DEFAULTS*
Hector Geffner, Judea Pearl
CSD-870066 $3.00

(*This work was supported in part by the National Science Foundation Grant, DCR 83-13875, IRI 86-10155.)
†This is a revised version of the report [Geffner et. al. 87]. The main departure here is the elimination of the notion of 'monotonicity in context' in favor of the more primitive notion of 'potential relevance' in section 2.3.)

A new system of defeasible inference is presented. The system is made up of a body of six rules which allow proofs to be constructed very much like in natural deduction systems in logic. Multiple extensions do not arise. Five of the rules are shown to possess a sound and clear probabilistic semantics that guarantees the high probability of the conclusion given the high probability of the premises. The sixth rule appeals to a notion of irrelevance; we explain both its motivation and use.

THE NON-AXIOMATIZABILITY OF DEPENDENCIES IN DIRECTED ACYCLIC GRAPHS
Dan Geiger
CSD-870048 $3.00

Recently, few models were proposed to capture the notion of relevance. Their main component consists of a mechanism to assign truth values to a 3-place relation I(x,z,y) where x,y,z are three non-intersecting sets of elements (e.g attributes or variables), and I(x,z,y) stands for the statement: "Knowing z renders x irrelevant to y." Among these models one can find the Probabilistic dependency model (Pearl & Paz [2]), the Undirected Graph (Pearl [1], Pearl & Paz [2]), Directed Acyclic Graph (Pearl [1]) and Hybrid Acyclic Graph (Pearl & Verma [3]). An important tool to investigate these models and their expressive power is a complete set of axioms. Such a set was found so far only for the UG model. In this paper, we show that there is no finite complete set of horn type axioms for DAGs. Moreover, we compare our result to a similar result in the Embedded Multi Valued Dependency (EMVD) model of relational database established by Parker and Kamran in 1980 ([4]), and independently by Sagiv and Walecka in 1982 ([5]). We point out that a stronger version of their incompleteness theorem can easily be proven for EMVDs. However, a similar extension for DAGs has not been fully established so far.
The paper is organized as follows. Section 2 reviews some previous work and the necessary terminology concerning dependency theory and the DAG model in particular. Section 3 presents the construction which shows that there is no finite complete set of horn type axioms for DAGs. Section 4 discusses extensions to the incompleteness theorem. Finally, section 5 summarizes the results and outlines the relation between dependency models and relational database theory.

A COMPLETE AXIOMATIZATION OF SOME DEPENDENCIES IN PROBABILISTIC MODELS
Dan Geiger
CSD-870056 $3.50

This paper deals with the task of establishing a complete axiomatic basis for the probabilistic independency relation "x is independent of y". This relation is shown to be an Armstrong relation. A set of axioms is presented and shown to be complete for this

relation. Some other dependencies in probabilistic models are presented for which a tractable membership algorythm is given. We also justify the use of Undirected Graphs as a represantation scheme for probabilistic dependencies.

OBJECT-ORIENTED SIMULATION OF POOL BALL MOTION
Arthur Paul Goldberg
CSD-870016 $16.50

This thesis presents a simulation model of pool ball motion. The model represents the positions of pool balls as they move around a pool table, colliding with each other, colliding with the sides of the table, and entering pockets.

The primary goal of our study has been to explore methods to *speed up* the pool ball simulation. The simulation techniques we use work towards this goal in two directions.

The main computational expense of a pool ball simulation is scheduling future ball collisions. Our first technique for increasing the speed of a simulation decreases the number of scheduled collisions by dividing the pool table area into *sectors*. We have implemented two pool ball simulation programs. In one the pool table area is continuous, while in the other the pool table area is divided into sectors. Experiments show that as the number of balls on the table increase the simulation runs considerably faster with a sectored pool table than with a pool table not divided into sectors.

Second, we have implemented the simulation to run on the *Time Warp* [Jefferson and Sowizral 82] distributed simulation system. This is the first large program to use Time Warp. It promises to be an important tool for studying the acceleration of simulation by distributed computing.

DISTRIBUTED SIMULATION AND THE TIME WARP OPERATING SYSTEM
David Jefferson, Brian Beckman, Fred Wieland, et. al.
CSD-870042 $7.00

This paper describes the Time Warp Operating System, under development for three years at the Jet Propulsion Laboratory for the Caltech Mark III Hypercube multiprocessor. Its primary goal is concurrent execution of large, irregular discrete event simulations at maximum speed. It also supports any other distributed applications that are synchronized by virtual time.

The Time Operating System includes a complete implementation of the Time Warp mechanism, and is a substantial departure from convential operating systems in that it performs synchronization by a general distributed process rollback mechanism. The use of general rollback forces a rethinking of many aspects of operating system design, including programming interface, scheduling, message routing and queueing, storage mangement, flow control, and commitment.

In this paper we review the mechanics of Time Warp, describe the TWOS operating system, show how to construct simulations in object-oriented form to run under TWOS, and offer a qualitative comparison of Time Warp to the Chandy-Misra method of distributed simulation. We also include details of two benchmark simulations and preliminary measurements of time-to-completion, speedup, rollback rate, and antimessage rate, all as functions of the number of processors used.

A MODELING METHODOLOGY FOR THE ANALYSIS OF CONCURRENT SYSTEMS AND COMPUTATIONS
Alex Kapelnikov*, Richard R. Muntz, and Milos D. Ercegovac
CSD-870061 $30.50

(*Presently with Unisys, Santa Monica, California)

In this paper, we describe a novel modeling methodology for evaluating the performance of distributed, multiple-computer systems. Our approach employs a set of analytic tools to obtain an estimate of the average execution time of a parallel implementation of a program (or transaction) in a distributed environment. These tools are based on an amalgamation of queueing network theory and graph models of program behavior. Hierarchical application of heuristic optimization techniques facilitates the analysis of large

and complex programs. A realistic example is used to illustrate the practical application of our methodology.

SYNTHESIS OF CUSTOM INTEGRATED CIRCUITS FROM A HIGH-LEVEL BEHAVIORAL DESCRIPTION
Steven Hennick Kelem
CSD-870028 $20.50

This dissertation descibes an innovative method for synthesizing error-free prototype integrated circuit masks from a behavioral-level algorithmic specification. The designer expreses algorithm and data types without having to specify a layout for their implementation or having to rely on time-consuming methods to obtain an implementation. The layout of the designed circuit is not explicitly specified by the designer, and it is determined from the algorithmic description. The circiuts are produced quickly, but not necessarily as compactly as can be done with exhautive, time-consuming, automatic or manual techniques. The circuits produced are properly formed (i.e. obey the fabrication design rules) so non-behavioral attributes such as area, speed and power can be estimated directly from these designs. When the data types in a design are changed, different operators to the new types are utilized and the non-behavioral attributes of the resulting circuit change. Comparisons of these attributes are the basis for choosing one design over the others.

This method of designing digital circuits is intended to meet the needs of a specialized set of problems, namely the design of special- purpose computing circuits. Such circuits typically are non-standard designs which require many iterations in the design process to obtain a cost-effective design. Therefore a rapid-prototyping design method, mapping algorithms given at a high level into circuit structures is highly desirable. This work is motivated by these needs and attempts to provide such a design tool.

SYSTEM ARCHITECT'S APPRENTICE (SARA) AS THE FOUNDATION FOR A METHODOLOGY-ORIENTED ADA® PROGRAMMING SUPPORT ENVIRONMENT†
Eduardo Aaron Krell
CSD-870001 $12.75

The design of inherently complex concurrent systems is inhibited by supportive languages and design environments. This dissertation probes the possibilities of integrating the strengths of a design method supported by a programming system called SARA (System ARchitect's Apprentice) and the well-known language, Ada®.

It has been recognized that one of the weaknesses of Ada Programming Support Environments (APSEs) is the lack of integration of design methods into them. It is shown that SARA and Ada can be integrated and, together, provide a significant advance over the state of the art.

An $o(n^2m^{1/2})$ DISTRIBUTED MAX-FLOW ALGORITHM
John Marberg, Eli Gafni
CSD-870002 $3.25

A distributed algorithm for finding maximum flow in a network is presented. The message complexity of the algorithm is $O(n^2m^{1/2})$, where n and m denote the number of nodes and the number of links in the network. This is an improvement by a factor of $\frac{n}{m^{1/2}}$ upon the previous best complexity.

A PROPOSAL FOR THE SYSTEMATIC DESIGN OF ARRAYS FOR MATRIX COMPUTATIONS
Jaime H. Moreno
CSD-870019 $6.25

We propose to develop a general and systematic methodology for the design of matrix solvers, based on the dependence graph of the algorithms. A fully--parallel graph is transformed to incorporate issues such as data broadcasting and synchronization, interconnection structure, I/O bandwidth, number and utilization of PEs, throughput, delay,

and the capability to solve problems larger than the size of the array. The objective is to devise a methodology which handles and relates features of the algorithm and the implementation, in a unified manner. This methodology assists a designer in selecting transformations to an algorithm from a set of feasible ones, and in evaluating the resulting implementations.

This research is motivated by the lack of an adequate design methodology for matrix computations. Standard structures (systolic arrays) have been used for these implementations, but they might be non--optimal for a particular algorithm. Reported systems have used ad--hoc design approaches. Some design methodologies have been proposed, but they do not address many important issues.

A preliminary version of the proposed methodology has been applied to algorithms for matrix multiplication and LU--decomposition. The approach produces structures which correspond to proposed systolic arrays for these computations, as well as structures which exhibit better efficiency than those arrays. The results show that different transformations on a graph may lead to entirely different computing structures. The selection of an adequate transformation is thus directed by the specific restrictions and performance objectives imposed on the implementation. The designer can identify and manipulate the parameters that are more relevant to a given application.

REMOVING ALGORITHM IRREGULARITIES IN THE DESIGN OF ARRAYS FOR MATRIX COMPUTATIONS
Jaime H. Moreno and Tomas Lang
CSD-870040 $4.75

We address some irregularities in matrix algorithms, as part of a systematic design methodology for arrays of processing elements (PEs) that we are researching. We propose a procedure to reduce the number of nodes in the fully--parallel dependence graph of matrix algorithms. We identify some irregularities which complicate such reduction of nodes and propose transformations to the dependence graph to remove them. The graphs obtained from such transformations are suitable for further transformations aimed towards regular algorithms or for direct implementation as arrays of PEs. The irregularities considered are bi--directional broadcasting and non--regular interconnection pattern between levels of the dependence graph. We use LU--decomposition, without and with pivoting, as examples of application of our transformations. The methodology results in triangular arrays for this computation, with better utilization than square arrays formerly proposed for it.

DAYDREAMING AND COMPUTATION: A COMPUTER MODEL OF EVERYDAY CREATIVITY, LEARNING, AND EMOTIONS IN THE HUMAN STREAM OF THOUGHT†
Erik Thomas Mueller
CSD-870017 $37.75

This dissertation presents a computational theory of *daydreaming:* the spontaneous human activity—carried out in a stream of thought—of recalling past experiences, imagining alternative versions of past experiences, and imagining possible future experiences. Although on the face of it, daydreaming may seem like a useless distraction from a task being performed, we argue that daydreaming serves several important functions for both humans and computers: 1) learning from imagined experiences, 2) creative problem solving, and 3) a useful interaction with emotions.

The theory is implemented within a computer program called DAYDREAMER which models the daydreaming of a human in the domain of interpersonal relations and common everyday occurrences. As input, DAYDREAMER takes English descriptions of external world events. As output, the program produces English descriptions of 1) actions it performs in the external world and 2) its internal "stream of thought" or "daydreams": sequences of events in imaginary past and future worlds.

Five major research issues are considered: 1) the generation and incremental modification of realistic and fanciful solutions or daydreams, 2) focusing attention in the presence of multiple active problems, 3) the recognition and exploitation of accidental relationships among problems, 4) the use of previous solutions or daydreams in

generating new solutions or daydreams, and 5) the interaction between emotions and daydreaming. DAYDREAMER consists of a collection of processing mechanisms and strategies which address each of the above issues: 1) a *planner*, a collection of *personal goals, daydreaming goals,* and *planning and inference rules* for the domain, and a *mutation* mechanism; 2) a control mechanism based on *emotions* as motivation; 3) a *serendipity* mechanism; 4) an *analogical planner* which stores, retrieves, and applies solutions or daydreams in a long-term *episodic memory;* and 5) mechanisms for initiating and modifying emotions during daydreaming and for influencing daydreaming in response to emotions. DAYDREAMER is able to generate a number of daydreams and demonstrate how daydreaming enables learning, creative problem solving, and a useful interaction with emotions.

LOG(F): A NEW SCHEME FOR INTEGRATING REWRITE RULES, LOGIC PROGRAMMING AND LAZY EVALUATION
Sanjai Narain
CSD-870027 $3.75

We present LOG(F), a new scheme for integrating rewrite rules logic programming and lazy evaluation. First, we develop a simple yet expressive rewrite rule system F* for representing functions. F* is non-Noetherian, i.e. an F* program can admit infinite reductions. For this system, we develop a reduction strategy called select and show that is possesses the property, select exibits a weak form of lazy evaluation. We then show how to implement F* in Prolog. Specifically, we compile rewrite rules of F* into Prolog clauses in such a way that when Prolog interprets these clauses it directly simulates the behavior of select. In particular, Prolog behaves lazily. Since it is not necessary to change already a logic programming system, a combination of rewrite rules, logic programming and lazy evaluation is achieved.

PARTIAL ORDER PROGRAMMING
D. Stott Parker
CSD-870067 $14.00

(This work supported by a State of California – IBM Los Angeles Scientific Center MICRO grant, "RAPPORT: Relaxation and Pattern-Oriented Rule Programming", and DARPA contracts MDA-903-82-C-0189 and F29601-87-K-0072.)

We introduce a programming paradigm in which statements are constraints over partial orders. A *partial order programming problem* has the form

$$\begin{array}{ll} \text{minimize} & u \\ \text{subject to} & u_1 \, B\!\geq v_1,\ u_2 \, B\!\geq v_2,\ \cdots \end{array}$$

where u is the *goal,* and $u_1 \, B\!\geq v_1,\ u_2 \, B\!\geq v_2,\ \cdots$ is a collection of constraints called the *program.* A solution of the problem is a minimal value for u determined by values for u_1, v_1, etc. satisfying the constraints. The domain of values here is a *partial order,* a domain D with ordering relation $B\!\geq$.

The partial order programming paradigm has interesting properties: It generalizes mathematical programming, dynamic programming, and computer programming paradigms (logic, functional, and others) cleanly, and offers a foundation both for studying and combining paradigms. It takes thorough advantage of known results for continuous functionals on complete partial orders, when the constraints involve expressions using only continuous and monotone operators. These programs have an elegant semantics coinciding with recent results on the relaxation solution method for constraint problems.

It presents a framework that may be effective in modeling of complex systems, and in knowledge representation for cognitive computation problems.

A FULL CHARACTERIZATION OF PSEUDOGRAPHOIDS IN TERMS OF FAMILIES OF UNDIRECTED GRAPHS*
Azaria Paz
CSD-870055 $4.50

*This work was supported by the National Science Foundation Grant #IRI-8610155.

Given a set of independencies closed under the pseudographoid axioms, an algorithm for

constructing a set of undirected graphs perfectly representing the given set of independencies is provided and its correctness is proved. Based on this algorithm, a full characterization of pseudographoids in terms of undirected graphs is given. A possible extension for full graphoids is investigated and some open problems are proposed. The algorithms introduced are illustrated with many examples. The main result of this report properly generalizes the I-mapness theorem in the [Pearl & Paz, 1986] report.

A PROBABILISTIC TREATMENT OF THE YALE SHOOTING PROBLEM*
Judea Pearl
CSD-870068 **$2.00**

(*This work was supported in part by NSF Grants #DCR 83-13875 and #IRI 86-10155.)

This paper uses the Yale Shooting episode as a test bed for presenting the probabilistic approach to default reasoning. Using a probabilistic interpretation of causality and irrelevance, the approach provides a powerful logic for constructing sound qualitative arguments.

DO WE NEED HIGHER-ORDER PROBABILITIES AND, IF SO, WHAT DO THEY MEAN?
Judea Pearl
CSD-870036 **$3.00**

(This work was supported in part by National Science Foundation Grant DCR 83-13875
†To appear in Proceedings of AAAI-87 Workshop on Uncertainty in AI, Seattle, Wash. July, 1987.)

The apparent failure of individual probabilistic expressions to distinguish uncertainty about truths from uncertainty about probabilistic assessments have prompted researchers to seek formalisms where the two types of uncertainties are given notational distinction. This paper demonstrates that the desired distinction is already a built-in feature of classical probabilistic models, thus, specialized notations are unnecessary.

AN INQUIRY INTO COMPUTER UNDERSTANDING*
Judea Pearl
CSD-870051 **$1.75**

*Submitted for the "Taking Issue" section of *Computational Intelligence*.
†This work was supported in part by NSF Grants #DCR 83-13875 and #IRI 86-10155.

Dr. Cheeseman has made a valuable contribution by compiling and articulating so forcefully the merits of probabilistic reasoning vis a vis deductive logic. The exposition is, in fact, so complete that I suddenly find myself in a strange desire to defend logic, a task I have not been trained to do, being myself an ardent student of probabilities.

PROBABILISTIC SEMANTICS FOR INHERITANCE HIERARCHIES WITH EXCEPTIONS*
Judea Pearl
CSD-870052 **$4.25**

*This work was supported in part by the National Science Foundation Grant, DCR 83-13875, IRI 86-10155.

Introduction

Let Γ be a collection of default statements of the form $I(p, q)$ where $I(p, q)$ means "p is typically a q" and $I(p, \neg q)$ reads "p is typically not a q".

Our task is to draw plausible conclusions from Γ. This requires that we establish a clear semantics for the meaning of each individual statement in Γ as well as for the absence of some statements NOT contained in Γ. For example, $\Gamma = \{I(a, bird), I(bird, fly)\}$ does not contain explicitly the statement $I(a, flies)$ neither $I(a, \neg flies)$ and, since every I allows exceptions, either one of the last two statements would be logically consistent with Γ. Yet, most people would regard the absence of $I(a, \neg flies)$ as a clue for the plausibility of $I(a, flies)$ but not vice versa; $I(a, \neg flies)$ might be accepted as a surprising fact but not as a conclusion.

The purpose of this report is to propose a probabilistic formulation that faithfully accounts for people's distinction between the plausible, the possible and the surprising. The formulation is offered as yet another standard for gauging the validity of proposed non-monotonic logics.

DECIDING CONSISTENCY IN INHERITANCE NETWORKS
Judea Pearl
CSD-870053 $2.50

Introduction

An *Inheritance Network* is a labeled directed graph, $\Gamma = (V, E^+ \ cup \ E^-)$, where the vertices V correspond to a set of atomic properties and E^+ and E^- are sets of positive and negative arcs, respectively. Positive arcs correspond to default rules of the type "p is a q" (e.g., "penguins are birds," "birds fly") and negative arcs correspond to default rules of the type "p is not a q" (e.g., "penguins do not fly").

An inheritance network is said to be *consistent* if there exists a probability model in which all the defaults are highly probable (Adams 1966, Pearl 1987). Formally, Γ is *consistent* if, for any $\varepsilon > 0$, there exists a probability function P such that:

$$P(q \mid p) \geq 1 - \varepsilon \quad whenever \quad (p, q) \ member \ E^+$$

$$P(q \mid p) \leq \varepsilon \quad whenever \quad (p, q) \ member \ E^-$$

Thus, consistent networks represent sets of default rules where conflicts can be explained away by appealing to rare exceptions, while inconsistent networks represent inexcusable contradictions, usually due to sloppy design or thoughtless inputs.

This report establishes the following criterion for deciding consistency: Γ is consistent iff for every pair of conflicting arcs $(p_1, q) \ member \ E^-$ and $(p_2, q) \ member \ E^+$, p_1 and p_2 must be distinct and there is no cycle of positive arcs that embraces both p_1 and p_2. The proof of this criterion is based on a theorem by Adams (1966) which provides a decision procedure for probabilistic consistency of general sets of formula.

Def - gAn assignment t of $\{0, 1\}$ values to the vertices of Γ is said to *falsify* a positive arc $(x, y) \ member \ E^+$ if $x=1$ and $y=0$ under t, and to *verify* $(x, y) \ member \ E^+$ if $x=1$, $y=1$ under t. A negative arc $(x, y) \ member \ E^-$ is falsified and verified by the respective assignments $(1, 1)$ and $(1, 0)$.

Def - gA set of arcs $E \ !subset \ E^+ \ cup \ E^-$ is said to be *confirmed* by a particular 0–1 assignment t, if no arc in E is falsified and at least one arc in E is verified under t.

Theorem (Adams, 1966) - g Γ is *consistent* if every non-empty subset of $E = E^+ \ cup \ E^-$ is confirmed by some assignment t.

Given a network Γ, our task is to devise a graph-theoretic criterion for deciding if Γ is consistent. Theorem 1 establishes such a criterion for the case where the positive arcs of Γ do not form a cycle while Theorem 2 extends the criterion to general networks.

POLYA'S "PATTERNS OF PLAUSIBLE INFERENCE" AND THE QUEST FOR MODULARITY*
Judea Pearl
CSD-870064 $2.00

(*This work was supported in part by the National Science Foundation Grants #DCR 83-13875 & 86-44931.)

Polya has proposed a list of patterns that characterize plausible reasoning. This paper investigates whether Polya's Patterns can be used as inference rules in reasoning systems. The answer seems to be negative, unless the rules are enriched with knowledge about relevance and causality.

THE LOGIC OF REPRESENTING DEPENDENCIES BY DIRECTED GRAPHS
Judea Pearl, Thomas Verma
CSD-870004 $2.75

Data-dependencies of the type "x can tell us more about y given that we already know z"

can be represented in various formalisms: Probabilistic Dependencies (PD), Embedded-Multi-Valued Dependencies (EMVD), Undirected Graphs (UG) and Directed-Acyclic Graphs (DAGs). This paper provides an axiomatic basis, called a *semi-graphoid*, which captures the structure common to all four types of dependencies and explores the expressive power of DAGs in representing various types of data dependencies. It is shown that DAGs can represent a richer set of dependencies than UGs, that DAGs completely represent the closure of their generating bases, and that they offer an effective computational device for testing consistency in a given set of dependencies as well as inferring new dependencies from that set. These properties might explain the prevailing use of DAGs in causal reasoning and semantic nets.

Science track
Major topic: Knowledge Representation
Subtopics: Data Dependencies, Logic of Relevance, Network Representations

PARALLEL ALGORITHMS AND ARCHITECTURES FOR BINARY IMAGE COMPONENT LABELING
Quoc Tuan Pham
CSD-870041 $16.25

We consider parallel algorithms in the PRAM (Parallel Random Access Machine) and SIMD (Single Instruction, Multiple Data Stream) models, and SIMD architectures for binary image component labeling, contour filling component shrinking and component counting. Current proposed SIMD architectures do not support efficient algorithms for the above three global operations (for images consisting of N pixels, the best time complexities of algorithms for component labeling are $O(N^{1/2})$ on the mesh-connected computer and $O(N^{1/4})$ on the pyramid computer). Current SIMD algorithms are complex, difficult to understand and highly dependent on the topology of the SMD interconnection network. We first give two O(logN) time, O(N) processor PRAM algorithms for the above operations. We then propose a general technique to map a certain type of PRAM algorithms into two proposed SIMD architectures with $O(\log^2 N)$ time penalty for PRAM concurrent read and concurrent write operations. In particular, we apply this technique to the above PRAM algorithms to obtain two $O(\log^3 N)$ time, O(N) processor SIMD algorithms which are more efficient than any other known SIMD algorithms for the above three operations.

THE RECOVERY OF CAUSAL POLY-TREES FROM STATISTICAL DATA
George Rebane and Judea Pearl
CSD-870031 $1.75

(This work was supported in part by the National Science Foundation Grant, DCR 83-13875.
†To appear in AAAI-87 Workshop on Uncertainty in AI, Seattle, Wash. July 1987.)

Poly-trees are singly connected causal networks in which variables may arise from multiple causes. This paper develops a method of recovering poly-trees from empirically measured probability distributions of pairs of variables. The method guarantees that, if the measured distributions are generated by a causal process structured as a poly-tree then the topological structure of such tree can be recovered precisely and, in addition, the causal directionality of the branches can be determined up to the maximum extent possible. The method also pinpoints the minimum (if any) external semantics required to determine the causal relationships among the variables considered.

FAULT-TOLERANT COMPUTING: ISSUES, EXAMPLES, AND METHODOLOGY
David Rennels
CSD-870024 $11.75

These lectures are intended to discuss a selection of fault-tolerant computers, with emphasis on issues and tradeoffs in the design of unmaintained systems which must provide dependable real-time operation over long-periods of time. Current research issues are discussed along with problems of developing and validating new high performance systems. The notes are compiled from previously published survey and research papers presented by this author. The information is modified and rearranged for tutorial purposes and augmented with additional presentation slides.

CASE STUDY: PERFORMANCE OF AN AEGIS-UNIX REMOTE FILE SYSTEM BRIDGE
Eve M. Schooler, Terence E. Gray
CSD-870063 **$1.75**

Remote File System Bridges (RFSB) focus on accommodating heterogeneity. They provide transparent access to remote files stored on computers running different operating systems. There are some general principles for architecting a RFSB, but each pair of operating systems provides unique challenges, and solutions for them do not always generalize. We attempt to evaluate the usefulness and robustness of a RFSB between an Apollo workstation running the Aegis operating system and a DEC/VAX 750 running the Berkeley UNIX 4.3 operating system. tdefine B>= % "⊒" %

DISTRIBUTION ANALYSIS OF PRODUCT FORM QUEUEING NETWORKS
Edmundo de Souza e Silva
CSD-870023 **$5.75**

We develop a new computational algorithm which computes joint queue length distributions for product form queueing networks with single server fixed rate, infinite server and queue dependent server service centers. Joint distributions are essential to calculate availability measures using queueing network modeling. This new algorithm is obtained using the physical interpretation of the main equation in the recently proposed MVAC algorithm. However, besides obtaining joint distributions, the new recursion has much simpler characteristics than the MVAC recursion.

QUEUEING NETWORK MODELS FOR LOAD BALANCING IN DISTRIBUTED SYSTEMS
Edmundo de Souza e Silva, Mario Gerla
CSD-870069 **$7.00**

(This research was supported by a grant from NSF INT-8514377 and CNPq-Brazil)

In distributed systems, load balancing can improve efficiency by migrating jobs from heavily loaded to lightly loaded sites. In this paper we present a method for optimal load allocation in a static environment. A queueing network model is used to evaluate response time; and mathematical programing techniques are used to find the load allocation that minimizes average response time. The method is not proposed as a substitute for dynamic, heuristic load balance policies; rather, it is perceived as a useful tool for resource allocation and capacity planning in distributed systems, and as a promising complement to dynamic policies in hybrid load balance strategies.

The method can handle very general classes of problems, including: distinct classes of jobs, multitasking within each job, and; jobs with spawned tasks. Several examples illustrating these applications are reported.

A NOTE ON THE COMPUTATIONAL COST OF THE LINEARIZER ALGORITHM FOR QUEUEING NETWORKS
Edmundo de Souza e Silva and Richard R. Muntz
CSD-870025 **$2.00**

Linearizer is one of the best known approximation algorithms for obtaining numeric solutions for product form queueing networks. In the original exposition of Linearizer, the computational cost was stated to be 0(MK sub 3) for a model with M queues and K job classes. We show in this note that with some straight forward algebraic manipulation Linearizer can be modified to require only
0(MK sub 2) computational cost.

We also discuss the space requirments for Linearizer and show that the space can be reduced to $0(MK)$ but with some increased computational cost.

HANDLING CONCEPTUAL AND STRUCTURAL AMBIGUITIES IN DIRECT MEMORY PARSING
Ronald A. Sumida, Michael Dyer, Margot Flowers
CSD-870018 **$2.75**

This paper discusses a method for dealing with conceptual and structural ambiguities in

natural language text. The approach presented in this paper features a constrained marker passing mechanism to search for both phrasal and role-level connections between words in the input, a short-term memory structure for storing these connections, and the incorporation of syntactic information using a phrasal approach. The short-term memory structure is used both to constrain search and to represent possibly conflicting interpretations of the input.

INDX, A SEMI-AUTOMATIC INDEXING PROGRAM
Kris Kiyoko Takata
CSD-870020 **$14.00**

Creating back-of-the-book indices is a difficult task involving intelligent and clerical processes. Programs have generally not achieved the level of intelligence required to perform the intelligent process of selecting terms for the index. Semi-automatic indexing programs perform the clerical process of preparing entries once the terms have been selected. The programs do not provide assistance in term determination and most require flooding the text with indexing commands. Indx differs from other semi-automatic indexing programs mainly because it does not require the insertion of indexing commands into the text to be indexed. The method by which indx assists in the creation of an index is introduced and compared with the characteristics of the other programs. This method includes the use of a program that aids the term determination process. The design, implementation, and application of indx are presented. Areas in which indx may be improved or enhanced are identified. An index of this thesis created with indx is included as an example.

ERROR RECOVERY IN MULTI-VERSION SOFTWARE[†]
Kam Sing Tso
CSD-870013 **$17.25**

In multi-version software (MVS), design faults are masked through the consensus of results from several diverse versions. Since we assume that all versions are likely to contain design faults, it is essential to be able to recover the state of individual versions as they fail.

A Community Error Recovery (CER) algorithm which makes use of the natural redundancy that exists among the diverse versions to recover failed versions has been designed. The CER algorithm is based on two levels of recovery: *Cross-check points*, which provide a consensus result for immediate masking and partial error recovery of the erroneous results, and *recovery points*, which provide a complete recovery of the erroneous states of failed versions. This provision of two levels minimizes both the disturbance to the system and the restrictions to the implementation of diverse versions. The CER recovery algorithm has been implemented on the UCLA DEDIX distributed MVS testbed. The purpose of DEDIX is to supervise and to observe the execution of N diverse versions of an application program functioning as a fault-tolerant MVS unit. DEDIX also provides a highly transparent interface to the users, versions, and the input/output system. The author's contributions to the cooperative DEDIX effort include the User Interface, Input/Output System, and the integration of the CER mechanism into the Version Layer, Local Executive, and Global Executive.

Markov models for reliability evaluation of CER have been developed and the results show that recovery may substantially improve the reliability of a MVS system. A large scale experiment is being conducted at UCLA in coordination with other institutions to determine the effect of fault tolerance techniques under carefully controlled conditions. To assess the effectiveness of the proposed CER algorithm, extensive testing were performed based on the five UCLA independently generated versions. The results of the evaluation are presented and discussed.

CAUSAL NETWORKS: SEMANTICS AND EXPRESSIVENESS
Thomas Verma
CSD-870032 $2.00

(This work was supported in part by the National Science Foundation Grant #DCR 85-01234.)

Dependency knowledge of the form "x is independent of y once z is known" can often be stored efficiently in graphical structures. Both strictly acausal (undirected) graphs [Pearl and Pa 1985], and strictly causal (directed acyclic) graphs [Pearl 1985] have been studied for this purpose. It is shown that the stratified protocol algorithm [Pearl 85] does in fact perfectly represent a dependency model with a DAG (directed acyclic graph) whenever possible. Further, for the class of semi-graphoids any DAG the algorithm generates is an I-map and the set of all of DAGs generated is a perfect map. Finally the idea of using graphs with both causal and acausal links is introduced and shown to be more expressive than the union of the previous two representations.

Science track

Major topic: Knowledge Representation

Subtopics: Data Dependencies, Logic of Relevance, Network Representations

DECOMPOSITION OF BOOLEAN FUNCTIONS ON A NETWORK OF POLYFUNCTIONAL NODES
Rik A. Verstraete & Jacques J. Vidal
CSD-870008 $9.75

The possibility of computing logic functions on networks of fixed topology but flexible functionality have received very little attention in the literature. Programmable logic modules were proposed in the sixties, but were found uneconomical under the technology of the times. With recent advances in VLSI, this may no longer be valid.

The limits of parallelism are reached when functions are computed combinatorially. A boolean network lies at the opposite extreme from a Turing machine, in a scale going from totally sequential to completely parallel. This report examines some of the problems associated with programming logic functions on distributed networks.

STRUCTURING CAUSAL TREE MODELS WITH CONTINUOUS VARIABLES
Lei Xu and Judea Pearl
CSD-870035 $2.25

(This work was supported in part by the National Science Foundation Grant, DCR 83-13875.
†To appear in AAAI-87 Workshop on Uncertainty in AI, Seattle, Wash. July 1987.)

This paper considers the problem of invoking auxiliary, unobservable variables to facilitate the structuring of causal tree models for a given set of continuous variables. Paralleling the treatment of bi-valued variables in [Pearl 1986], we show that if a collection of coupled variables are governed by a joint normal distribution and a tree-structured representation exists, then both the topology and all internal relationships of the tree can be uncovered by observing pairwise dependencies among the observed variables (i.e., the leaves of the tree). Furthermore, the conditions for normally distributed variables are less restrictive than those governing bi-valued variables. The result extends the applications of causal tree models which were found useful in evidential reasoning tasks.

TECHNICAL PAPERS

On Style, Expressibility, and Efficiency in Functional Programming Languages

Gabriel Robins

Computer Science Department
University of California, Los Angeles

Abstract

A functional style of programming was proposed by Backus as an alternative to conventional programming paradigms. Functional programming is intended to alleviate what Backus dubbed as the "Von Neumann Bottleneck," a restrictive set of biases deeply ingrained in the thought processes of programmers. The functional programming paradigm is investigated here through the actual implementation of a series of common algorithms. One of the novel features of this paradigm is the lack of side-effects; it is shown how this property can be used to achieve considerable efficiency improvement in arbitrary implementations of functional languages, using the idea of *value-caching*. This scheme is then shown to reduce the execution of certain computations from exponential-time to polynomial-time, a formidable speedup. It is also shown that the expressive power of functional languages can be greatly enhanced via the utilization of both *macros* and *idioms*, without changing the original semantics of the language.

1. Introduction

A functional style of programming was proposed in 1978 by John Backus [Backus] as an alternative to conventional programming paradigms. Functional programming is intended to alleviate what Backus dubbed as the "Von Neumann Bottleneck," a restrictive set of biases which by now is deeply ingrained in the thought processes of programmers. Backus argues quite eloquently that because the processor and the memory of a conventional computer are connected via a narrow bus, processors end up spending most of their time ferrying useless data to and from the main memory.

The more serious problem, however, is that the classical "word-at-a-time" model of computer design has over the years managed to permeate up through the abstraction levels and become fused into most programming languages, and indeed into the very thought processes of the majority of computer scientists. According to Backus, this phenomenon is directly responsible for many horrendously complex programming languages with convoluted and even inconsistent semantics, where it is often impossible to prove formally even the most trivial correctness properties.

Backus gave a syntax and semantics for a candidate functional programming language. Some of the novel features of his language are the lack of variable names, assignment statements, global state, and side-effects. Instead, several powerful combining operators can be used to compose functions, yielding clean and simple semantics, conciseness of expression, and easily-derivable formal correctness properties. See [Vegdahl] for a more detailed discussion of these issues. Considerable work regarding FP has also been performed at UCLA: [Alkalaj], [Alkalaj, Ercegovac, and Lang], [Arabe], [Meshkinpour], [Meshkinpour and Ercegovac], [Patel and Ercegovac], [Patel, Schlag, and Ercegovac], [Pungsornruk], [Sausville], [Schlag 86], [Schlag 84], [Worley].

In this paper I investigate some of the claims made by proponents of functional programming, through the actual implementation of a series of common algorithms. Based on these examples, I then draw some conclusions with regards to the practical advantages and flaws of the functional programming paradigm. The language I use is Berkeley's functional programming language, FP [Baden]. All FP forms which appear in this paper were verified to be correct by actually typing them into the FP v. 4.2 (4/28/83) system. The rest of the paper assumes that the reader is familiar with Berkeley FP; for a brief refresher on the syntax and semantics of FP, the reader is referred to the appendix.

I show how the lack of side-effects can be used to achieve considerable efficiency improvement in arbitrary implementations of functional languages, using the idea of *value-caching*. This scheme is proved to reduce the execution time of certain FP computations from exponential to polynomial, a remarkable speedup. Finally, it is shown that the expressive power of functional languages can be greatly enhanced via the utilization of both *macros* and *idioms*, without changing the original semantics of the language.

2. FP Solutions to Some Common Problems

In this section we examine how various common problems and algorithms can be handled in FP. We do this in order to get some feeling as to what does "typical" FP code look like. The notation "X ==> Y" will be used to denote that when the form X is evaluated it yields the value Y, as in the following: "+ @ [%1 , %2] ==> 3".

2.1. Permutations

The following function produces a permutation of the second argument, according to the order specified by the first argument. That is, given the following data vector of length N:$<d_1,d_2, \ldots , d_N>$, and a permutation vector $<p_1, p_2, \ldots , p_N>$, where $1 \le p_i \le N$ and $p_i \ne p_j$ if $i \ne j$, we would like to return the vector $<d_{p_1}, d_{p_2}, \ldots , d_{p_N}>$. The following FP function will accomplish this task:

{PERM1 &pick @ distr}

For example:

PERM1 : <<3 1 2 5 4> <is going what here on>> ==> <what is going on here>

But now one might argue that any straight-forward implementation of FP would cause the "distr" operator to "copy" its second argument a number of times proportional to the length of the first argument, using $O(N_2)$ space in total. In attempting a response to such criticism we may modify our definition as follows:

{PERM2 (null @ 1 -> [] ; apndl @ [pick @ [1 @ 1, 2] , PERM2 @ [tl @ 1, 2]])}

PERM2 : <<2 4 3 1> <permuted 4 get things>> ==> <4 things get permuted>

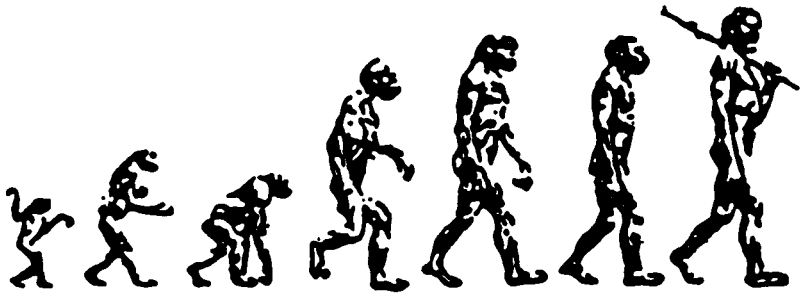
Now we are using tail-recursion to accomplish the same task. In constructing the seemingly "more space-efficient" function PERM2, we were presumptuous in assuming that the underlying FP implementation allocates space and executes FP forms in a certain way, and regardless of how close to the truth our assumptions may have been, these are the very kinds of considerations Backus is trying to free us from in the first place.

In programming in FP we must give up many familiar and useful concepts, such as variable names, a state, side-effects, etc. But we gain many pleasant properties in return for our "sacrifice": conciseness, more easily provable correctness, better parallelism, and freedom from details. If during FP programming we worry about efficiency and implementation issues, the efficiency of our code will become implementation-dependent, and our minds will be shackled by the very details from which we escaped when we sought refuge from the "Von Neumann bottleneck" in the first place.

If we are to enjoy the full benefits of a functional programming system, we must delegate most of the efficiency and implementation issues to the underlying (optimizing) compilers. One of the main benefits of FP is that it has very clean semantics, yielding relatively painless correctness proofs, and it is amenable to simple and powerful algebraic manipulations and transformations. Such algebraic transformations could be used to automatically convert/compile an FP program into a version which would take advantage of the inherent parallelism or other special features of the architecture we happen to be executing on.

In summary, I am *not* arguing that efficiency considerations are not important; I am simply

arguing that it is more the responsibility of the machines to catch up with functional paradigms of thinking and not the burden of functional programming to accommodate archaic architectures. The situation is dramatized in the following diagram:



Functional Programming

Hardware Support

2.2. List Reversal

Suppose we wanted to reverse a list. What first comes to mind is a recursive definition for the reversal operation, namely that the reversal of a list is equal to the last element followed by the reversal of the rest of the list. In FP this could be expressed as follows:

{REV1 (null -> id; apndl @ [last,REV1 @ tlr])}

For example:

REV1 @ iota : 13 ==> <13 12 11 10 9 8 7 6 5 4 3 2 1>

In trying to determine whether any parallelism is inherent in list reversal, we try to apply the highly-parallel "tree-insert" FP operator, as in the following:

{REV2 ! [2,1]}

REV2 @ iota : 13 ==> <<<13 <12 11>> <10 <9 8>>> <<7 <6 5>> <<4 3> <2 1>>>>

We observe that this simple function is almost what we want, except for the extra levels of list depth, so we revise our definition as follows:

{REV3 ! (concat @ [(atom @ 2 -> [2] ; 2) , (atom @ 1 -> [1] ; 1)])}

REV3 @ iota : 13 ==> <13 12 11 10 9 8 7 6 5 4 3 2 1>

Now we have obtained a highly parallel, yet simple and concisely-expressed function to reverse a list. I consider this to be functional programming at its best.

2.3. Membership in a List

Suppose we needed a function that given an element and a list, would determine whether that element appears in that list. Our first attempt is the following:

```
{MEMB ! or @ &= @ distl}

MEMB  : <13 <3 6 4 13 98 -45 13 73>>    ==>    T
```

The idea here is to distribute the search key over the entire list and "or" together the results of all the comparisons of these pairs. Now suppose we know ahead of time that our list was sorted. Using the classical binary-search technique, we could speed up the computation to "require" at most O(log N) time by recursively searching half-lists, as follows:

```
{BS (null @ 3 -> = @ [1,1 @ 2];
  (<= @ [1,1@rotr@2] -> BS @ [1,2]; BS @ [1,3])) @ apndl @ [1,split @ 2]}
```

Note that although this "optimization" appears to reduce the evaluation time, this may not really happen in reality. For example, a close examination of the code for BS would reveal that we have assumed that the operation "split" operates in constant time. If indeed split is implemented in such a way as to require time linear in the size of its argument, BS may in fact run slower than its seemingly "less efficient" counterpart MEMB.

This example again illustrates our previous argument that FP programmers should not try to "second-guess" the implementation, because they may loose. It is conceivable that in the future intelligent compilers could algebraically manipulate such expressions and, for example, replace a linear search with a binary search whenever it finds a monotonicity property to be preserved. So again, I claim that such "optimizations" should be left to the compilers.

Of course one must draw the line somewhere. Given that the problem of whether two programs are equivalent or not is undecidable, in the general case it is not possible to perform these optimizations automatically. Yet even undecidability results do not mean that special classes of instances could not be handled automatically, as well as efficiently. For example, it is probably too much to hope that a compiler would recognize an O(N5)-time max-flow algorithm and replace it with an equivalent but more efficient O(N3) time algorithm; on the other hand it is not far-fetched to expect that an intelligent compiler could determine that a certain piece of code behaves like a look-up table and consequently implement it as a hash-table rather than a list, yielding better efficiency.

2.4. Fibonacci Numbers

The Fibonacci numbers are defined as the sequence of integers beginning with two 1's with each subsequent integer being the sum of the previous two. This recursive definition immediately yields an FP program:

```
{F (<=@[id,%2] -> %1; +@[F @ D,F @ D @ D])}   # returns the i^th Fibonacci number
{D -@[id,%1]}                                 # decrements by 1
{FIB &F @ iota}                               # returns the first n Fibonacci numbers

FIB : 18  ==>  <1 1 2 3 5 8 13 2134 55 89 144 233 377 610 987 1597 2584>
```

Although this program is both elegant and concise, it is quite "inefficient", because any conventional implementation of it would force the re-computation of the function on many of the same arguments. We can alleviate this flaw as follows:

```
{FIB2 tlr @ tlr @ FF @ [iota , [%1,%1]]}

{FF (null @ 1 -> 2 ; FF @ [tl@1 , apndr @ [2, + @ [last@2 , last@tlr@2]]])}
```

Now successive Fibonacci numbers are appended to a growing list, where the last two elements

are used directly to compute the next Fibonacci number. This optimization is essentially an algorithmic one and reduces the "execution" time from exponential-time to polynomial-time. a considerable savings. I believe such programmer-affected optimization to be quite valid, and also one which FP programmers should endeavor to concern themselves with. Later in this paper, we describe a method of achieving considerable running-time improvement while preserving the functional appearance of functions such as the Fibonacci numbers function; this is accomplished via the idea of value-caching. In other words, sometimes we can "have the cake and eat it too," functionally speaking.

2.5. Sorting

Sorting a list entails rearranging its elements in such a way as to preserve some monotonicity property with respect to successive elements. For simplicity, we assume that our elements are integers and that the monotone property we wish to sort with respect to is the normal arithmetic "less-than" or "<". Our first sorting algorithm is fashioned after the classical *selection-sort*:

```
{SS (null -> [] ; apndl @ [1, SS @ SUB] @ [MIN,id])}
{SUB concat @ &(= -> [] ; [2]) @ distl}
{MIN ! (< -> 1 ; 2)}

MIN : <3 5 0 9 213 -7 1>          ==>   -7
SUB : <13 < 2 13 a 5 w 13>>       ==>   <2 a 5 w>
SS  : <4 7 5 1 0 9 11 7>          ==>   <0 1 4 5 7 9 11>
```

The function MIN determines the least element in a given list, while the function SUB returns a new list without any occurrences of a given element. Sorting is accomplished by finding the least element, putting it first, followed by the sorted list of the remaining elements.

Next we try another method for sorting, which is similar to our selection-sort algorithm. To sort a list, we keep rotating it until the first element is least. We then continue this process on the tail of the list, until the entire list is sorted. We call this algorithm *rotation-sort*:

```
{RS (null -> []; (= @[1,MIN] -> apndl @ [1, RS @ tl]; RS @ rotr))}
{MIN ! (< -> 1 ; 2)}
RS : < 5 3 13 -2 -13 13 0 99 213>  ==>  <-13 -2 0 3 5 13 13 99 213>
```

As an interesting aside, our list is now allowed to have duplicates and such duplicate elements are not eliminated from the list. Although our rotation-sort algorithm is specified very concisely, it does not exhibit much parallelism and would require "quadratic" time to run in any conventional implementation. Another method of sorting entails swapping adjacent pairs of elements until the list becomes monotonic. We call this scheme *odd-even-sort*:

```
{SORT (while UNORD (apndl @ [1,SWAP @ tl] @ SWAP))}
{UNORD lor @ &> @ trans @ [tlr,tl]}
{SWAP concat @ &FLIP @ pair}
{FLIP (null @ tl -> id; (> -> [2,1]; id))}

UNORD : <-5 2 5 9>                ==> F
FLIP : <13 2>                     ==> <2 13>
SWAP : <76 13 0 12 33 22 0 13>    ==> <13 76 0 12 22 33 0 13>
SORT : <76 13 -12  22 0 13 213 0 >  ==> <-12 0 0  13 13 22 76 213>
```

The idea here is to look at the pair of elements 1 and 2, 3 and 4, etc, and swap each pair in-place if they are out of order. During the second pass, we similarly treat the pairs 2 and 3, 4 and 5, etc. We keep repeating this two-phased pass over the data until the list becomes sorted. Note the high degree of inherent parallelism here, since all the pairs may be examined/swapped in parallel,

A common method for sorting with a good average-time behavior is quicksort. The quicksort algorithm selects a pivot element and splits the input into two sublists, one consisting of elements less than (or equal-to) the pivot, and the other list consisting of elements greater-than the pivot

element. Each of these two sublists is similarly sorted in a recursive fashion, and the final result then consists of the first list, followed by the pivot element, followed by the second list. The FP implementation of *quicksort* follows:

```
{QS (<= @ [length,%1] -> id; concat @ [QS @ L, [1], QS @ R])}
{L concat @ & (>          -> tl; []) @ distl @ [1,tl]}
{R concat @ & (<=-> tl; []) @ distl @ [1,tl]}

L : <7 3 9 -4 13 27>                    ==> <3 -4>
R : <7 3 9 -4 13 27>                    ==> <9 13 27>
QS : <76 13 -12  22 0 13 55 213 0 >     ==> <-12 0 0  13 13 22 55 76 213>
```

Again, we note the conciseness and elegance of the FP specification. Moreover, it is well-known that quicksort has an O(NlogN) average-time behavior (although the normal worst-case behavior is still quadratic). Later in this paper we describe how the FP definition of quicksort can be made even more concise using macros.

2.6. Prime Numbers

A prime number is a positive integer with *exactly* two distinct divisors: itself and 1. We would like to specify a functional program which given an integer N, returns a complete list of primes between 2 and N, inclusively. Here is our first pass at this problem:

```
{PRIMES concat @ &(PRIME -> [id] ; []) @ tl @ iota}
{PRIME land @ &(not @ = @ [%0,id]) @ &mod @ distl @ [id,tl@iota@-@[id,%1]]}

PRIMES : 20     ==> <2 3 5 7 11 13 17 19>
```

The idea here is to process all the integers between 2 and N, trying to determine which ones have no factors other than themselves and 1. All integers satisfying this property are primes and are therefore returned as a list. The function PRIME is a predicate which tests a single integer for primality, while the function PRIMES maps PRIME onto an entire list of candidates for primes.

We note however, that to test for primality, we need only try to divide by factors no greater than the square root of our candidate integer, since if our integer is divisible by a factor which is greater than the square root, the result of the division is less than the square root and so we would have tried that integer earlier. Furthermore, other than the integer 2, only odd integers need be considered for primality, and only odd integers (up to the square-root of the candidate) need be considered as factors, since an even integer can not divide an odd one. These "optimizations" are reflected in a new version of our prime number generator:

```
{PRIMES apndl @ [%2, concat @ &(PRIME -> [id] ; []) @ ODDS]}
{PRIME land @ &(not @ = @ [%0,id]) @ &mod @ distl @ [id,ODDS @ SQRT]}
{SQRT 1 @ (while (< @ [*@[id,id]@1, 2]) [+@[1,%1],2]) @ [%1,id]}
{ODDS  apndl@[%2,&(+@[*@[id,%2],%1])
                        @iota@-@[+@[/@[id,%2],mod@[id,%2]],%1]]}

PRIMES : 50     ==>   <2 3 5 7 11 13 17 19 23 29 31 37 41 43 47>
```

Note that while this scheme is "faster" than the previous one, the code is beginning to look rather ugly. In fact, further speedup is possible since when checking for the primality of a candidate integer, we need only to consider other primes (up to the square-root of the candidate) as possible factors. The lesson here again is that the more "efficient" a functional program is, the messier it tends to become.

In the case of prime numbers, for example, I think that only the first "optimization" (the one about going only up to the square root) is of the type with which the functional programmers need concern themselves, because it is "algorithmic" in nature, and relies on a non-trivial number-theoretic fact; it so happens that it is also the only one which increases the "efficiency" of our

functional program by a non-constant factor.

We give another algorithm for prime generation, the *sieve of Erastostenes*:

```
{SIEVE concat @ &(MEMB -> [];[1]) @ distr @
                [id,concat @ &&* @ &distl @ distr @ [id,id]] @ tl @ iota}
{MEMB !or @ &= @ distl}

SIEVE : 18      ==>  <2 3 5 7 11 13 17>
```

The idea here is to list all the integers and erase all factors of 2, all factors of 3, and so on. Everything which remains after this process of elimination is by necessity a prime. Actually, in this implementation we do the opposite: we form all possible products and then consider as prime all the "missing" integers. Again we see an example of the trade-off between concise specification and computational "efficiency".

3. FP vs. APL

There are some notable similarities between APL and FP is quite similar in spirit and syntax to APL. Some functions and operators in FP are taken directly from APL; for example, the FP iota function is equivalent to the APL iota function ("ι"). Similarly, the FP operator "right-insert" ("!") is equivalent to the APL operator "reduce" ("/"). The difference with respect to these operators is that in FP one can compose any number of operators together in a concise and semanticaly clean manner, while in APL only a pre-defined limited set of types of compositions are allowed.

Both in FP and APL complex operations may be defined very concisely, often giving rise to "one-line" programs. Yet unlike FP, the semantics of APL are very complex, being muddled by state, local and global variables, parameter passing, operator overloading, goto statements, and the existence of several hundreds of different functions and operators. I have seen one-line APL programs which take the better part of an hour to understand; a possible moral here is that "absolute power corrupts absolutely." While it is true that some pieces of FP code may also seem rather dense, the situation in FP is still much better than in the notorious APL "one-liners".

4. FP vs. LISP

FP is also similar in many respects to LISP. Both languages rely heavily on lists as their primary data structures, and both languages provide a rich set of list-manipulation primitives. On the other hand, LISP functions are more defined than their FP counterparts; for example in FP **tl : <>** is undefined, while in LISP **(CDR NIL)** is still **NIL**. Given that FP functions are bottom-preserving, it is often the case that an FP function would return "?" on a boundary or pathological datum, while the LISP equivalent would still return the "right" answer.

Here is an example of when it is preferable to have functions which are more defined on "boundary conditions". Given an atom x and a list of atoms y, we would like to return the set of all atoms in y which directly follow occurrences of x in y. The obvious way of specifying this in FP elegantly is:

```
{F concat @ &(= @ [2,1@1] -> [2@1] ; []) @ distr @ [trans @ [tlr,tl] @ 2 , 1]}

F : <3 <m 3 4 5 6 3  3 x 3>>        ==>    <4 3 x>
F : <y <>>                          ==>    ?
```

The problem here of course is that our function is not defined for the pathological case when the second argument is the empty list, while it would be preferable to define the answer in that case to be the empty list. We could modify our original function to check for that boundary condition as a special case, but if the FP functions **tl** and **tlr** were originally designed to return the empty list when called upon the empty list, our function F would not have the problem mentioned here. For example, if **tl, tlr,** and **trans** were redefined thus:

```
{TL (= @ [ length , %0 ] -> [] ; tl )}
```

```
{TLR (= @ [ length , %0 ] -> [] ; tlr )}
{TRANS (land @ &null -> [] ; trans)}
```

F could now use these "more defined" specification to work correctly in all cases, including the boundary conditions:

```
{F concat @ &(= @ [2,1@1] -> [2@1] ; []) @ distr @ [TRANS @ [TLR,TL] @ 2 , 1]}

F : <y <x>>      ==>    < >
F : <y <>>       ==>    < >
```

The point we wish to stress is that had some of the primitive functions been just a little more "defined", the programmer would not have had to worry about these boundary conditions, and freeing the programmer from such details is clearly a goal of the functional-thinking paradigm.

5. Macros in FP

FP currently does not allow the use of macros. This is, however, an artificial restriction since macros can be specified and used in a way that will not alter the original semantics of FP whatsoever. FP macros specify simple "static" substitution rules that would "expand" certain pieces of FP code. In other words, a macro-expansion preprocessor would be invoked before execution ever commences.

To accommodate macros, the original FP syntax would be extended as follows:

```
macro-definition ==> '#define' name '(' argument-names ')' macro-body
name             ==> letter (letter | digit | '_')*
argument-names   ==> name ( ',' name)*
macro-body       ==> (function-form | name)*
function-form    ==> simple-function | composition | construction
                         | conditional | constant | insertion | alpha
                         | while | '(' function-form ')' | macro-expansion
macro-expansion  ==> name '(' arguments ')'
arguments        ==> string ( ',' string)*
string           ==> (letter | number | special-character)*
```

It is understood that a macro expansion form and the corresponding macro-definition must agree on the number of arguments. It is further observed that arguments to macros may include arbitrary strings, not just FP objects, as long as a valid FP form is obtained when macro-expansion is terminated. Moreover, macro-expansion is a recursive process, where macros may refer to other macros, as long as termination of the macro-expansion process is guaranteed.

As an example, the macro *#define add1(x) + @ [x,%1]* would expand the form *add1(k)* into the form *+ @ [k,%1]*. As a second example of using macros, our previous definition of quicksort could be specified more concisely as:

```
#define S(X) concat @ &(X -> tl; []) @ distl @ [1,tl]
{QS (<= @ [length,%1] -> id; concat @ [QS @ S(>=), [1], QS @ S(<)])}
```

The above form, after the macro-expansion into "pure" FP, will become:

```
{QS (<= @ [length,%1] -> id; concat @ [QS @
                              concat @ &(>= -> tl; []) @ distl @ [1,tl], [1], QS @
                              concat @ &(< -> tl; []) @ distl @ [1,tl]])}
```

Note that in this example, we can not achieve the same effect by simply defining a new function, because the argument we really need to pass is a functor itself, not a value. In general, macros will increase the readability of FP code by making the FP representation of a function more concise, whenever certain pieces of code are identical or at least very similar. A macro can then be used to "factor out" that common piece of code, so that it does not have to be specified or

expressed more than once; in particular, macros would allow the definition of an APPLY template, similar in spirit to the LISP apply function, something that is impossible to implement in "pure" FP.

5.1. The Implementation of Macros

We emphasize that the usage of macros as defined here does not extend nor change the original semantics of FP in any way. In fact, an implementation of this idea might consist of a macro-expansion pre-processor which would take as input a specification in macro-enhanced FP (that is, an FP program with some macros) and output an equivalent "pure" FP program after properly doing the requisite macro expansion and substitution. The result would then be fed into the original FP processor.

For example, to implement a macro pre-processor for Berkeley FP, one could extend the original FP syntax to accommodate macros as described above, and construct a UNIX lex(1) and yacc(1) specification to lexically analyze and parse the new syntax. The associated production rules would perform the required macro expansion, producing as output an equivalent FP program in the original "pure" syntax. The function "yyparse" can then be used as a "filter" to FP.

An easier alternative to using heavy-artillery such as yacc(1) and lex(1) consists of the following scheme: since many existing languages already have rather fancy macro-preprocessors, why not "fool" such a tool into doing our FP macro-processing work for us? For example, the standard UNIX C compiler has such a macro-preprocessor, and in fact it may be invoked without the actual C compilation. So an implementation of this scheme would consist of the following two steps:

1) Run the file containing "macro-enhanced" FP program through the C macro preprocessor and collect the resulting "pure" FP output in a file. This is accomplished by specifying the -E flag: **cc -E macro-prog.fp > pure.fp**

2) Run the resulting file as usual through the normal FP system: **fp < pure.fp**

I have already tested this scheme with several examples, including the quicksort example stated above, and it worked flawlessly. In fact, if one combines the above two steps into one UNIX command aliased to an appropriate name, the entire macro-expansion process would be transparent to the user. Of course, if the format of our proposed FP macros deviates at all from that of C, this trivial scheme would fail and we would be forced to revert back to our direct-translation scheme using lex(1) and yacc(1).

Macros could also be used to "modify" the syntax of FP by providing alternate ways of expressing FP constructs; such extensions to FP in no way alter the original semantics of FP but simply provide some "syntactic sugar" to ease the programming task.

6. The Need for Idioms

I found that during FP programming, many constructs repeat themselves. For example, the function to decrement a counter, to sort, etc. Such common and repeatedly used functions can and should be provided for in the implementation of an FP-like language. Indeed such functions can be easily specified within FP itself, thereby not adding to the original set of "primitives", but instead be provided as a *library* of functions or macros.

I claim that it would be very convenient to define certain idioms within FP. Note that this convenience is not in conflict with the fact that such idioms may be easily specified using other FP forms. For example, the original definition of FP still contains the idiom "length", even though "length" may easily be defined as a recursive application of **tl**, or perhaps even more cleverly as:

{length | + @ &%1}

Perlis and Rugaber have already devised such a set of idioms for APL [Perlis and Rugaber]. I believe that a similar set should be identified and defined for FP. In particular, I recommend the following list of possible idioms as a starting-point for the further development of this idea:

Removal of duplicates from a list - given a key x and a list y, return a new list z which is a copy of y except that all elements equal to x have been removed.

Sorting a list given a specified partial-order predicate - given a list, return the list sorted according to a specified (or default) comparison predicate. Since the said predicate is an argument to our "sort" routine, an implementation using macros would probably be best here.

Generalized 'iota' - given A, B, and C, return a sequence of all multiples of C, beginning with A and no larger than B.

Vector element uniqueness - given a sequence of objects, return "true" if there exists a repeated element in this sequence, or "false" otherwise.

Is X a permutation of Y? - given two sequences x and y, determine whether x is a permutation of y. This may be accomplished by sorting both into some "canonical order" and then comparing.

Minimum/maximum element - find the minimum (or maximum) element in a given sequence, using a specified min (or max) function, or else use a default comparison function.

Membership in a vector/set - test whether a given element x is a member of vector/set y.

Union of vectors/sets - given two vectors/sets x and y, return a vector/set of elements which are either in x or in y, without duplicates.

Intersection of vectors/sets - given two vectors/sets x and y, return a vector/set of elements which are both in x and in y, without duplicates.

Subtraction of vectors/sets - given two vectors/sets x and y, return a vector/set of elements which are in x but not in y, without duplicates.

Subset of a vector/set - given two vectors/sets x and y, determine whether each element of x is also an element of y.

Transitive closure - given a sequence of pairs or objects, return the transitive-closure of this set, interpreting the pairs as a specification of a binary relation.

Math functions - square root, exponentiation, factorials, binomial coefficients, successor, predecessor, etc.

Flatten a list - return a list of all atoms in a given list, disregarding the original level (or nesting-depth) of each.

7. Improved Efficiency Through Value-Cashing

We have observed that there exists a sharp trade-off in certain computations between conciseness-of-expression and "computational efficiency." For example, in the case of generating Fibonacci numbers, the elegant version of the algorithm appears as follows:

```
{F (<=@[id,%2] -> %1; +@[F @ D,F @ D @ D])}   # returns the i^th Fibonacci number
{D - @[id,%1]}                                # decrements by 1
```

We also observed earlier that alternative ways of specifying the Fibonacci function yield more complex and difficult to read specifications. So the question now is how efficiently can the current specification be used to compute the i^{th} Fibonacci number on a conventional architecture, and whether the situation can be improved. We settle the former part of the question first, and argue that a conventional implementation would require exponential time to compute the current Fibonacci numbers specification.

It is well-known that the N^{th} Fibonacci number is given by the expression:

$F_n=[\phi^n-(1-\phi)^n]/(2\phi-1)$ where $\phi=(1+\sqrt{5})/2$

The constant $\phi\approx1.618033989$ is also known as the golden ratio, and when N approaches infinity, the term $(1-\phi)^n$ approaches zero and therefore F_n approaches $\phi^n/(2\phi-1)$ but since ϕ is a constant, we have $F_n=\Theta(\phi^n)$ or that F_n grows exponentially. Since a conventional implementation can not be expected to solve such recurrence relations in closed-form, it must use recursion to compute F_n. But the problem is that F will be recomputed many times at the same values, and since the value of F_n must be obtained from the lowest levels of the recursion via repeated additions of the constant 1, the total number of additions will be equal to the value of F_n, or proportional to ϕ^n. It follows that a naive computation of the Fibonacci sequence from the recursive specification will require exponential time.

Of course, in order to remain within polynomial-time, we could explicitly remember old values, but this is contrary to the spirit of the functional paradigm. We would like to combine the elegance of the concise recursive specification with the "efficiency" of the "optimized" version.

A hybrid can actually be achieved through what I call *value-caching*. The idea is to save argument/value pairs for certain difficult-to-compute functions, so that those values could be looked up in the future instead of having to be recomputed. We define a new FP operator @$ (read compose-cache) to have the exact same semantics as the familiar FP operator @, except that after the composition is computed, the result is stored away in such a way that next time the same composition needs to be performed, the result is already known and can be used immediately without any recomputation or further delay. Clearly, the new composition operation so defined has the exact same semantics as the normal composition operator, and moreover, this entire scheme remains transparent to the programmer.

Because in FP there are no side-effects, composing a function f with an argument g will yield the exact same result at *any* stage of the computation, and that is why old results will *always* be valid in the future. This scheme can be implemented using hash-tables to store the value/result pairs for each function we would like to thus speed up. When a function is "called" upon an "argument", the argument is looked up in the hash-table associated with that function, and if a result is found it is returned immediately. If not, the result is computed from scratch, and then it is saved in the hash-table using the argument as a key, whereupon the result is passed on to the rest of the computation.

Of course not all functions should be so treated, only those that are non-trivial to compute. For example, using this scheme, our Fibonacci function would now appear as follows:

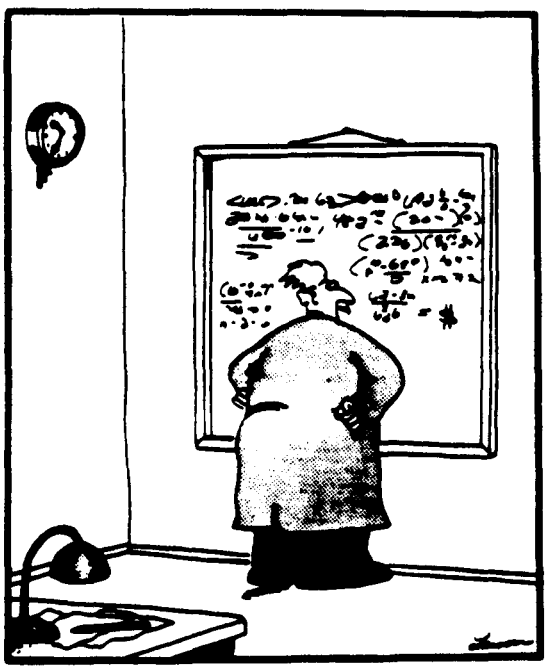
```
{F (<=@[id,%2] -> %1; +@[F @$ D,F @$ D @ D])}      # returns the i^th Fibonacci num
{D - @[id,%1]}                                     # decrements by 1
```

Notice that we have not lost any of the conciseness or elegance apparent in our original specification. And assuming the cache/lookup process takes near-constant time, the Fibonacci function now will run in near-linear time on any conventional implementation so enhanced, an exponential improvement!

This scheme will work for arbitrary functional languages, not just FP. Moreover, the @$ operator need not be used manually by the programmer, but simply be invoked automatically by the implementation whenever it is determined that doing so is likely to improve the efficiency. Some possible criteria for automatically performing value-caching are when considerable time has been spent inside some particular function that has relatively short input and output, when a particular function is being called a very large number of times on only a small set of different values, and when a function (or a set of functions) is highly recursive.

All of these situations can be detected automatically by the implementation, so it is possible to automate the idea of speed-up by value-caching. Sometimes, however, it may be the case that the programmer can better determine where caching should and should not be performed, so it may be desirable to give the programmer a "manual-override" capability with respect to value-caching. I propose two new variations on the composition operator, and a slightly modified behavior of the old composition operator:

@$ - value-caching is *always* performed.
@^$ - value-caching is *never* performed.
@ - architecture automatically decides when cashing should be performed.



Einstein discovers that time is actually money

Other instances where value-caching will yield considerable increase in execution efficiency are applications where computations have a geometric meaning in Euclidean D-space. For example, Conway's game of "life", where a cellular automata in the 2-dimensional plane is simulated and where the "next-generation" rule is a function of the local neighborhood surrounding the cell. A straight-forward recursive computation to determine the state of a given cell would run in time exponential in the magnitude of the generation to be computed, where the base corresponds to the size of the neighborhood around cells. On the other hand, the same algorithm running in a value-caching environment would run in polynomial time.

Generally speaking, whenever a dynamic-programming algorithm of some kind is used, new values are dependent on certain old values (with respect to some set of indices), an therefore an ordinary functional solution would likely to give rise to an inefficient execution. But if value-caching is used in the implementation, the execution time of the exact same algorithm is likely to improve drastically.

Recently it has come to my attention that the idea of caching values in order to avoid recomputations is also discussed in [Keller and Sleep]. They discuss several ways of storing and using old values in order to improve execution efficiency, and also address the problem of "purging" or keeping the cache size from growing without bounds during a computation. They also give as example another application area where value-caching would greatly speed up the execution time, namely in the finite-element method.

8. Conclusion

Through a series of examples I have investigated programming style and practical considerations

in FP. I have demonstrated that it is often the case in FP that concise functions become verbose and complicated when certain "efficiency optimizations" are attempted. I have also argued that certain so-called "optimizations" may really not be optimizations at all, and should not be undertaken by an FP programmer.

I have briefly discussed the similarities between FP and APL and LISP, and argued that certain FP functions should be "more defined" and that there is a need for some additional FP "primitive" functions or macros. I described a scheme where macros can be implemented in a clean way, leaving the original semantics of FP intact. I have argued that FP would benefit from a library of idioms, or commonly-used constructs upon which the programmer can draw.

It is possible via novel architectures and intelligent compilers to support efficient FP-like languages, where many optimizations can take place automatically, leaving the programmer free to see much of the high-level picture without drowning in the details. I introduced one such possible optimization, value-caching, and showed that it can reduce the execution time of certain computations by as much as an exponential amount.

9. Acknowledgments

I would like to thank Professor Milos Ercegovac for introducing me to FP, and for the numerous illuminating discussions which inspired this paper. Warm thanks go to Professor Sheila Greibach for encouraging me to polish up this paper and submit it to the quarterly; she made numerous thoughtful comments, and also helped in unsplitting my infinitives.

10. Bibliography

Alkalaj, L., "A Uniprocessor Implementation of FP Functional Language", MS Thesis, *UCLA Computer Science Department TR CSD-860064*, April 1986.

Alkalaj,L., Ercegovac, M., and Lang, T., "A Dynamic Memory Management Policy for FP", *1987 Hawaii International Conference on Systems Science.*

Arabe, J., "Compiler Considerations and Run-Time Storage Management for a Functional Programming System", Ph.D. Dissertation, *UCLA Computer Science Department, CSD TR 86004*, August 1986.

Backus, J., "Can Programming Be Liberated from the Von Neumann Style? A Functional Style and its Algebra of Programs", *Communications of the ACM*, Vol 21, No. 8, August, 1978, pp. 613-641.

Baden, S., *Berkeley FP User's Manual*, Rev. 4.1, September 29, 1987.

Keller, R., Sleep, R., "Applicative Caching: Programmer Control of Object Sharing and Lifetime in Distributed Implementations of Applicative Languages", *Proceedings of the 1981 ACM Conference on Functional Programming Languages and Computer Architectures*, October, 1981, pp. 131-140.

Meshkinpour, F., "On Specification and Design of Digital Systems Using and Applicative Hardware Description Language", M.S. Thesis, *UCLA Computer Science Department, Technical Report No. CSD-840046*, November 1984.

Meshkinpour, F., and Ercegovac, M., "A Functional Language for Description and Design of Digital Systems: Sequential Constructs", *IEEE Proc. of the 22nd ACM/IEEE Design Automation Conference*, 1985, pp.238-244.

Patel, D., and Ercegovac, M., "A High-Level Language for Hardware-Oriented Algorithm Design," presented at the *Symposium Electronics and Communications in the 80's*, Indian Institute of Technology, Bombay, February 1983.

Patel, D., Schlag, M., and Ercegovac, M., "vFP: An Environment for the Multi-Level

Specification, Analysis, and Synthesis of Hardware Algorithms," *Proc. 1985 ACM Conference on Functional Programming Languages and Architectures*, Nancy, France, Springer-Verlag Lecture Notes 201, 1985.

Perlis, A., Rugaber, S., "The APL Idiom List", *Research Report #87*, Yale University.

Pungsornruk, S., "A Threaded FP Interpreter/Compiler", M.S. Thesis, *UCLA Computer Science Department TR CSD-860061*, March 1986.

Sausville, M., "Gathering Performance Statistics on Hardware Specified in FP", *UCLA Computer Science Department Internal Report*, March 20, 1986.

Schlag, M., "Layout from a Topological Description", *Ph.D. Dissertation*, UCLA Computer Science Department, Summer 1986.

Schlag, M., "Extracting Geometry from FP Expressions for VLSI Layout", *UCLA Computer Science Report CSD-840043*, October 1984.

Vegdahl, S., "A survey of Proposed Architectures for the Execution of Functional Languages", *IEEE Transactions on Computers*, Vol c-33, No. 12, December, 1984, pp. 1050-1071.

Worley, J., "A Functional Style Description of Digital Systems", M.S. Thesis, *UCLA Computer Science Department, Technical Report CSD-860054*, February 1986.

10. Appendix: A Brief Overview of FP

In order to keep this paper self-contained, we briefly review the syntax and semantics of FP; for a more detailed description of FP, see [Baden].

The notation "X ==> Y" is used to denote that when the form X is evaluated it yields the value Y; this is simply a notational convenience. FP objects consist of numbers, strings, atoms, and lists; examples are %213, "hi there", FOO, <I am <a list>>, and <>. An integer constant must be preceded by a percent sign, to distinguish it from the primitive selector functions. We use the form "F : x" to denote the result of the function F applied to the argument x; for example, + : <%2 %3> ==> 5. The unique error atom (or "bottom") is denoted as "?". In what follows, the formal FP specifications are also explained informally for clarity.

$$\mathbf{K} : x \Longrightarrow \text{if } x=\langle x_1, x_2, \cdots, x_n\rangle \text{ and } 1\le k\le n \text{ then } x_k \text{ else } ?$$

For every integer K there exists a primitive selector function **K** that returns the K^{th} element of a list. For example:

3 : <fi fy foo fum> ==> foo

Other selector primitives:

pick : $x \Longrightarrow \text{if } x=\langle y,\langle x_1, x_2, \cdots, x_n\rangle\rangle \text{ and } 1\le y\le n \text{ then } x_y \text{ else } ?$

last : $x \Longrightarrow \text{if } x=\langle\rangle \text{ then } \langle\rangle \text{ else if } x=\langle x_1, x_2, \cdots, x_n\rangle \text{ and } 1\le n \text{ then } x_n \text{ else } ?$

first : $x \Longrightarrow \text{if } x=\langle\rangle \text{ then } \langle\rangle \text{ else if } x=\langle x_1, x_2, \cdots, x_n\rangle \text{ and } 1\le n \text{ then } x_1 \text{ else } ?$

tl : $x \Longrightarrow \text{if } x=\langle y\rangle \text{ then } \langle\rangle \text{ else if } x=\langle x_1, x_2, \cdots, x_n\rangle \text{ and } 2\le n \text{ then } \langle x_2, x_3, \cdots, x_n\rangle$ else ?

tlr : $x \Longrightarrow \text{if } (x=\langle\rangle) \text{ or } (x=\langle y\rangle) \text{ then } \langle\rangle \text{ else if } x=\langle x_1, x_2, \cdots, x_n\rangle \text{ and } 2\le n$ then $\langle x_1, x_2, \cdots, x_{n-1}\rangle$ else ?

pick selects the y^{th} element out of a given list, **last** selects the last element, first selects the **first** elememt, **tl** chops the first element off and returns the rest (like the LISP cdr function), and **tlr** chops the last element off and returns the rest. Examples are:

pick : <2 <a b c>>	==>	b
last : <w x y z>	==>	z

first : <<1 2> <3 4 5> <6 7 8 9>> ==> <1 2>
tl : <who are you> ==> <are you>
tlr : <fi fy fo fum> ==> <fi fy fo>

Structural and list-manipulation primitives:

distl : x ==> if x=<y,<>> then <>
else if x=<y,<x_1,x_2, ··· ,x_n > > and 1≤n then < <y,x_1 >,< y,x_2 >, ··· ,< y,x_n > > else ?
distr : x ==> if x=<<>,y> then <>
else if x=<<x_1,x_2, ··· ,x_n>,y>and1≤nthen<<x_1,y>,<x_2,y>, ··· ,<x_n,y >> else ?
apndl : x ==> if x=<y,<>> then <y>
else if x=<y,< x_1,x_2, ··· ,x_n>> and 1≤n then <y,x_1,x_2, ··· ,x_n> else ?
apndr : x ==> if x=<<>,y> then <y>
else if x=<<x_1,x_2, ··· ,x_n>,y> and 1≤n then <x_1,x_2, ··· ,x_n,y> else ?
trans : x ==> if x=<<x_1,x_2, ··· ,x_n>,<y_1,y_2, ··· ,y_n>, ··· ,<z_1,z_2, ··· ,z_n>> and 1≤n
then <<x_1,y_1, ··· ,z_1>, ··· ,<x_n,y_n, ··· ,z_n>> else ?
reverse : x ==> if x=<> then <> else if x=<x_1,x_2, ··· ,x_n> and 1≤n
then <x_n,x_{n-1}, ··· ,x_1> else ?
rotl : x ==> if x=<> then <> else if x=<y> then <y> else if x=<x_1,x_2, ··· ,x_n> and 1≤n
then <x_2,x_3, ··· ,x_n,x_1> else ?
rotr : x ==> if x=<> then <> else if x=<y> then <y> else if x=<x_1,x_2, ··· ,x_n> and 1≤n
then <x_n,x_1,x_2, ··· ,x_{n-2},x_{n-1}> else ?
concat : x ==> if x=<<x_1,x_2, ··· ,x_n>,<y_1,y_2, ··· ,y_m>, ··· ,<z_1,z_2, ··· ,z_1 > > and 1≤n,m, ··· ,1
then <x_1,x_2, ··· ,x_n,y_1,y_2, ··· ,y_m, ··· ,z_1,z_2, ··· ,z_1> else ?
pair : x ==> if x=<x_1,x_2, ··· ,x_n> and 1≤n is even then < <x_1,x_2>, ··· ,<x_{n-1},x_n>>
else if x=<<x_1,x_2, ··· ,x_n> and n is odd then <<x_1,x_2>, ··· ,<x_n>> else ?
split : x ==> if x=<y> then <<y>,<>> else if x=<x_1,x_2, ··· ,x_n> and 1≤n
then <<x_1,x_2, ··· ,$x_{floor(n/2)}$>,<$x_{floor(n/2)+1}$, ··· ,x_n>> else ?

distl (**distr**) pairs from the left (the right) the first (second) argument with each element of the second (first) argument, **apndl** (**apndr**) appends the first (second) element to the end of the list given by the second (first) argument, **trans** treats the argument lists as the rows of a matrix and transposes this matrix, **reverse** reverses a list, **rotl** (**rotr**) does a left (right) circular shift on a list, **concat** concatenates two lists, **pair** pairs up the elements in a list, and **split** divides a list into nearly equal halves. Examples follow:

distl : <u <r m 2>> ==> <<u r> <u m> <u 2>>
distr : <<i u voo> doo> ==> <<i doo> <u doo> <voo doo>>
apndr : <<hi di> ho> ==> <hi di ho>
apndl : <r <2 d 2>> ==> <r 2 d 2>
trans : <<1 2 3> <a b c>> ==> <<1 a> <2 b> <3 c>>
reverse : <to be or not to be> ==> <be to not or be to>
rotl : <10 20 30 40> ==> <20 30 40 10>
rotr : <<1 2> <3 4> <x y z>> ==> <<x y z> <1 2> <3 4>>
concat : <<x> <y z> <u q t>> ==> <x y z u q t>
pair : <romeo juliet samson delila> ==> <<romeo juliet> <samson delila>>
split : <hair1 hair2 h3 h4 h5> ==> <<hair1 hair2 h3> <h4 h5>>

Predicates:

atom : x ==> if x is an atom then **T** else if x≠? then **F** else ?
null : x ==> if x =<> then **T** else if x≠? then **F** else ?
length : x ==> if x=<> then 0 else if x = < x_1 ,x_2 , ··· ,x_n > then n else ?

atom tests whether the argument is an atom, null tests whether it is the empty list, and **length** returns the length of a list. Examples follow:

atom : firefox ==> **T**
null : <not empty at all> ==> **F**
length : <tell how long <am I>> ==> **4**

Arithmetic and logical operations:

+ : x ==> if x=<a,b> and a and b are numbers then a+b else ?
(and similarly for –,*, /, <, >, ≤, ≥, =, ≠)
xor : x ==> if x=<a,b> and a and b are boolean then xor(a,b)else ?
(and similarly for **and, or, not, nand,** and **nor**)

These operators compute the standard arithmetic and logical functions. Examples follow:

+: <2 3>	==>	5
–: <4 13>	==>	-9
*: <13 11>	==>	143
/: <81 3>	==>	27
<: <100 999>	==>	T
=: <k k>	==>	T

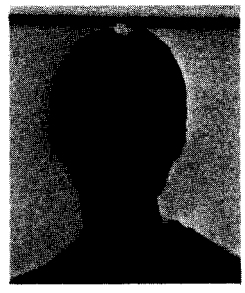
Functional forms, combinators, and conditional:

f1 @ f2 : x ==> f1 : (f2 : x)
[$expr_1$,$expr_2$, · · · ,$expr_n$] : x ==> <$expr_1$:x,$expr_2$:x, · · · ,$expr_n$:x>
& expr : x ==> if x = <> then <> else if x=<x_1,x_2, · · · ,x_n>and1≤n
then <expr : x_1, expr : x_2, · · · ,expr : x_n > else ?
! expr : x ==> if x = <y> then <y> else if x=<x_1 ,x_2 , · · · ,x_n > and 2≤n
then expr : <x_1,!expr:<x_2, · · · ,x_n> >else ?
\expr : x ==> if x = <y> then <y> else if x=<x_1 ,x_2 , · · · ,x_n > and 2≤n
then expr : < expr : <x_1 , · · · ,x_{n-1} >,x_n > else ?
| expr : x ==> if x = <y> then <y> else if x=<x_1 ,x_2 , · · · ,x_n > and 2≤n
then expr : < | expr : <x_1 , · · · ,$x_{floor(n/2)}$ > , | expr : < $x_{floor(n/2)}$ > > else ?
(C -> A ; B) : x ==> if C : x - <> then A : x else if C : x = <> then B : x else ?

@ denotes functional composition and is by far the most common FP operator. Square brackets ([and]) denote list construction. The functor & maps the given function onto each element of the argument list. !, \, and | invoke several kinds of parallel tree evaluation, while the construct (C -> A;B) is the FP conditional operator: if C evaluates to true, A is evaluated and returned, otherwise B is evaluated and returned. Examples follow:

* @ tlr : <3 2 1>	==>	6
[+@[2,2],1,2] : <b 1>	==>	<2 b 1>
& atom : <yes <no> <<maybe>>>	==>	<T F F>
! * : <1 2 3 4 5 6>	==>	720
| id : <1 2 3 4 5 6>	==>	<<<1 2> 3> <<4 5> 6>>
(atom -> %1 ; %-1) : oxygen	==>	1

ABOUT THE AUTHOR



Gabriel Robins was born in New York in February of 1962, and was raised in Israel. In 1974 he moved back to New York, and in 1976 he moved to Los Angeles, where he is currently residing. He received his B.S. in Mathematics and Computer Science from UCLA in 1983 and his M.S.E. in Computer Science from Princeton University in 1985. For the last several years he has been working part-time for Information Science Institute at Marina Del Rey, where he has contributed to the development of the knowledge representation language NIKL [1] and has most recently designed and implemented a portable software tool for the pictorial display of graphs [2,3,4], now commercially marketed by ExperTelligence Inc. Gabriel truly enjoys teaching, and is now a teaching assistant in the UCLA CS department, for the automata theory and formal languages course. In addition, he has developed with Dr. Eli Gafni a computational geometry course, given at UCLA for the first time in the Winter of 1988. Gabriel's interests include algorithms, data structures, complexity, graph theory, formal languages, computational geometry, user interfaces, and distributed & parallel algorithms. Presently he is working with Dr. Eli Gafni on some aspects of distributed protocols.

[1] Kaczmarek, T., Bates, R., and Robins, G., "Recent Developments in NIKL", *Proceedings of the Fifth National Conference on Artificial Intelligence*, American Association of Artificial Intelligence, Philadelphia, August 1986.

[2] Robins, G., "The ISI Grapher: a Portable Tool for Displaying Graphs Pictorially", invited talk in *Symboliikka '87*, Helsinki, Finland, August 17-18, 1987 (reprinted in Multicomputer Vision, Levialdi, S., Chapter 12, Academic Press, London, 1988).

[3] Robins, G., "The ISI Grapher Manual", *ISI Technical Manual/Report ISI/TM-88-197*, USC/Information Sciences Institute, Marina Del Rey, February 1988.

[4] Robins, G., "Applications of the ISI Grapher", *Proceedings of the Artificial Intelligence and Advanced Computer Technology Conference*, Long Beach, California, May 1988.

STATEMENT OF PRINCIPLES

The *Computer Science Department Quarterly* is issued three times a year by the Computer Science Department of the University of California at Los Angeles. It is directed to prospective students, current students, and alumni of the Computer Science Department as well as to interested members of the academic and scientific community. The purpose of the Quarterly is two-fold: (1) to present in a systematic form information regarding faculty resources, academic programs, research activities, and administrative matters relevant to the Computer Science Department; (2) to present short technical contributions, including tutorial articles and reports on recent research within the Computer Science Department.

In order to fulfill the first of the above objectives, each of the four annual issues emphasizes a different aspect of the Computer Science Department activities.

> The Fall/Winter quarter issue is devoted to a description of the faculty resources of the Computer Science Department including Visiting Scholars, Teaching Assistants, Research Staff, and Fellowship Holders, with attention to special honors and awards received by faculty members, bibliographical citations, and lists of graduate degree recipients during the preceding calendar year.
>
> The Spring quarter issue emphasizes academic programs, stressing particularly the requirements for the M.S. and Ph.D. degrees as well as descriptions of Computer Science Department course offerings and recommended course groupings.
>
> The Summer quarter issue includes brief descriptions of the research interests and projects of the Computer Science faculty and reports on some of the sponsored and unsponsored research programs within the Computer Science Department.

Computer Science Department
School of Engineering and Applied Science
3731 Boelter Hall
University of California
Los Angeles, CA 90024-1596