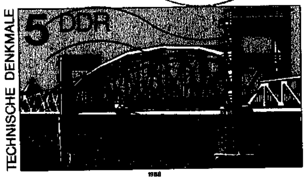
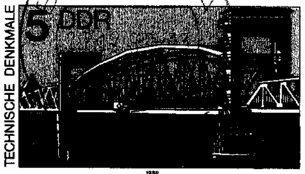
Dr. rer. nat.
R. Karl-Adolf Zech
Diplom-Mathematiker
Schliemannstraße 28
Berlin
DDR – 1058
German Democratic Republik

oben

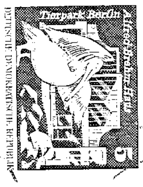
DRUCKSACHE





BERLIN BP
1005

Dres. Arnie Dyck, Jay Black,
Kelly Booth
University of Waterloo
Dept. Computer Science
WATERLOO
ONT N2L 3G1
CANADA



BERLIN BP
1055

Dr. rer. nat.
R. Karl-Adolf Zech
Diplom-Mathematiker

Schliemannstraße 28
Berlin
DDR – 1058 German Democratic Republic

Dear colleagues:

I should be very obliged to you for sending me a free copy of your paper entitled:

CS-89-07: Delivering a better introductory course in -- please.

sent Feb. 19/90

Thanking you in advance Yours sincerely,

[illegible]

UNIVERSITY OF DENVER
(Colorado Seminary)
DENVER, COLORADO

ENDORSEMENT OF CHECK CONSTITUTES A RECEIPT IN FULL FOR ITEMS LISTED

455548

INV. DATE	INV. NO./DESCRIPTION	P.O. NO.	VCH. NO.	GROSS AMOUNT	DISC. AMOUNT	NET AMOUNT
12/29/89	CS-89-07 (3 COP		0003523	6.00		6.00 E
				6.00		6.00

Rec'd payment & sent reports Jan. 17/90

VENDOR NO.	T0110700070	DATE	01/10/90	CHECK NO.	19-455548

DETACH CHECK BEFORE CASHING OR DEPOSITING

CS-89-07 - DELIVERING A BETTER INTRODUCTORY COURSE IN COMPUTER SCIENCE

AUTHORS: Arnie Dyck, Jay Black, Kelly Booth

$6.00

Willy, please get 3 of these [illegible]

ABSTRACT:

In revising the introductory computer science curriculum at the University of Waterloo, a number of general issues arose. The first was the need to integrate many fundamental topics of computer science into a small number of courses, to provide an overview of the field for students a variety of potential majors. The second was the requirement that the time spent by students (especially programming time) be in proportion to the weight of the course within the overall curriculum. Finally, we wished to close the perceived gap between "computer literacy" courses (where experience with state-of-the-art desktop applications predominates) and more traditional computer science courses (where the emphasis is on system design and implementation). The first issue is reflected in the choice of a "spiral" approach that covers theoretical and practical topics in computer science. The elimination of "open-ended" assignments in favour of time-limited laboratory sessions with preparatory tutorials complements the lecture presentations both with hands-on experience using commercial application packages and with programming assignments where students implement and test their own algorithms. In designing and implementing the new curriculum it became clear that the logistics of delivering the course material were at least as important to success as were the choice of the material itself. A support team of faculty, staff, graduate student teaching assistants, and undergraduate tutors provides a diverse skill set that has proven to be a cost-effective delivery system for the new curriculum.

PRICE: $2.00

CS-89-08 - DESIGNING AND MODELING VLSI SYSTEMS AT THE

REGISTER-TRANSFER LEVEL

AUTHOR: Farhad Mavaddat

ABSTRACT:

In this paper we discuss some of the properties that a design abstraction must have, and make suitable proposals for the RT abstraction. To this effect, we propose a small set of design primitives with well-defined behaviours, and give rules for their composition, all within the framework of a strongly-typed signal environment known as the SDC Model of register-transfer design. The primitives are chosen to permit a mathematical treatment of designs, but they are also highly design-oriented, and provide the designer with a useful and friendly set of building blocks. We also present user-friendly graphical and textual interfaces to the model. To prove our designs correct, we use Algorithmic State Machines (ASM) as the frameworks for analyzing and reasoning about an SDC-based design.

PRICE: $2.00

Send To : Univ of Denver
Dept of Math/computer Sci.
Denver, Co 80208

University of Waterloo
Department of Computer Science
Waterloo, Ontario N2L 3G1
Research Reports 1989 (January to April)

CS-89-01 - RESTRICTED-ORIENTED CONVEX SETS

AUTHORS: Gregory J.E. Rawlins, Derick Wood

ABSTRACT:

A *restricted-oriented convex* set is a set of points whose intersection with any line, in a given set of orientations, is either empty or connected. This notion generalizes both orthogonal convexity and normal convexity.

The aim of this paper is to establish a mathematical foundation for the theory of restricted-oriented convex sets. To this end, we prove the restricted-oriented analogs of some basic properties of convex sets and also present a decomposition theorem for them.

Keywords: convex sets, convex hulls, restricted-orientation convexity, computational geometry.

PRICE: $2.00

CS-89-02 - A DECOMPOSITION THEOREM FOR CONVEXITY SPACES

AUTHORS: Gregory J.E. Rawlins, Derick Wood

ABSTRACT:

We utilize the unifying framework of families of convexity spaces for the treatment of various notions of planar convexity and the associated convex hulls. Our major goal is to prove the refinement and decomposition theorems for families of convexity spaces. These general theorems are the applied to two examples: restricted-oriented convex sets and *NESW*-convex sets. The applications demonstrate the usefulness of these general theorems, since they give rise to simple algorithms for the computation of the associated convex hulls of polygons.

Keywords: convexity, convexity spaces, geometry, convex hulls, decomposition, refinement.

PRICE: $2.00

UNIVERSITY OF VICTORIA

P.O. BOX 1700, VICTORIA, B.C., CANADA V8W 2Y2
TELEPHONE (604) 721-7209, TELEX 049-7222, FAX (604) 721-7292

DEPARTMENT OF COMPUTER SCIENCE

Dec 10, 1989

Please send me a copy of CS-89-07 "Delivering a better introductory course in computer science" by Dyck, Black and Booth.

With many thanks,

M. H. van Emden

M. H. van Emden
Dept. of Computer Science
University of Victoria
Victoria, B.C. V8W 2Y2 Canada

sent
Jan. 4/90

csuvic@uvunix.bitnet

csuvic@uvunix.uvic.ca

Telephone: (042) 27 0555
Telex: 29022
Fax: (042) 27 0477

THE UNIVERSITY OF WOLLONGONG

NORTHFIELD AVENUE — WOLLONGONG

MAILING ADDRESS: P.O BOX 1144
WOLLONGONG, N.S.W.
AUSTRALIA, 2500

PAGE: 1
ORDER No. 086562
(QUOTE ON ALL DOCUMENTS)

UNIVERSITY OF WATERLOO
DEPT OF COMPUTER SCIENCE
WATERLOO ONTARIO
CANADA N2L 3G1

ORDER DATE 11/12/89
CURRENCY: $ US

G.S.D. Cont. Item No.	QTY.	UNIT	DESCRIPTION	RATE	UNIT	DISC. %
	2	EA	CS-89-07 DELIVERING A BETTER INTRODUCTORY COURSE IN COMPUTER SCIENCE	2.00	EA	
			TERMS C.W.O. TOTAL US	4.00		

sent Jan. 4/90

THESE GOODS ARE FOR THE EXCLUSIVE USE OF THE UNIVERSITY OF WOLLONGONG AND ARE NOT FOR RESALE AND ARE EXEMPT FROM SALES TAX UNDER ITEM 63A SALES TAX (EXEMPTIONS AND CLASSIFICATIONS) ACT.

FOR THE UNIVERSITY OF WOLLONGONG
[illegible]
AUTHORISED OFFICER DATE 13/12/89

ACCOUNT CODE	AMOUNT	JOB No.	REQ. No.	SUPPLIER No.	DESCRIPTION
031433006323	5		31475	409324	RESEARCH REPORTS

PLEASE OBSERVE THE FOLLOWING INSTRUCTIONS

1. DELIVER OR ADDRESS PACKAGES AS FOLLOWS: CENTRAL STORE, ATTENTION COMPUTING SCIENCE
   THE UNIVERSITY OF WOLLONGONG, NORTHFIELD AVENUE, WOLLONGONG, N.S.W., 2500, AUSTRALIA
2. QUOTE ORDER No. 086562 ON YOUR DELIVERY DOCKET AND YOUR INVOICE.
3. PACK THE DELIVERY DOCKET WITH THE GOODS. **DO NOT** PACK YOUR INVOICE WITH THE GOODS.
4. SEND YOUR INVOICE AND STATEMENT OF ACCOUNT TO: THE FINANCE OFFICER, THE UNIVERSITY OF WOLLONGONG.
5. FOR ROAD TRANSPORT USE "IPEC" ON FREIGHT COLLECT BASIS.
6. THIS ORDER IS ISSUED TO YOU SUBJECT TO THE PRICES STATED HEREIN. VARIATION IN PRICE, QUALITY OR QUANTITY MUST BE APPROVED BY THE SUPPLY OFFICER PRIOR TO DELIVERY.
7. TECHNICAL ENQUIRIES SHOULD BE MADE TO:—

CS-89-13 - SOLVING SYSTEMS OF ALGEBRAIC EQUATIONS,

OR THE INTERFACE BETWEEN SOFTWARE AND MATHEMATICS

AUTHORS: Gaston H. Gonnet, Michael B. Monagan

ABSTRACT:

This paper describes two fundamental aspects of the solution of systems of algebraic equations: the use of *substitution* and the use of a *complexity function* to determine, at each step, which of several methods /equations /unknowns to use. These techniques, which have received little attention in the past, can be viewed as in the interface between mathematics and software. An implementation of these techniques in *Maple* $^{Cha\,83}$ is described. Timing results comparing the Maple, *Macsyma* $^{Mos\,74}$ and *Reduce* $^{Hea\,71}$ symbolic algebra systems on a range of sample problems are presented. The proposed method proves to be superior to other techniques.

PRICE: $2.00

CS-89-14 - REASONING ABOUT FUNCTIONAL DEPENDENCIES

GENERALIZED FOR SEMANTIC DATA MODELS

AUTHOR: Grant E. Weddell

ABSTRACT:

We propose a more general form of functional dependency for semantic data models that derives from their common feature in which the separate notions of *domain* and *relation* in the relational model are combined into a single notion of *class*. The feature manifests itself in a richer terminological component for their query languages in which a single term may traverse any number of properties (including none). We prove the richer expressiveness of this more general functional dependency, and exhibit a sound and complete set of inference axioms. Decision procedures are developed that apply when the dependencies included in a schema correspond to keys, or when the schema itself is acyclic. The theory is then extended to include a generalization of select-join queries. Of particular significance is that the queries become an additional source of functional dependency constraints. Finally, we outline several applications of the theory to various problems in physical design and in query optimization. The applications derive from an ability to predict when queries can have at most one solution.

PRICE: $2.00

If you would like to order individual reports please forward your order, along with a cheque or international bank draft payable to the **Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, to the Research Report Secretary.**

If you wish to receive all reports published in 1989 our **Subscription Rate** is $125.00 Canadian. Please refer to the above paragraph for payment instructions.

Please indicate your current mailing address.

MAILING ADDRESS:

MR. JOHN FULCHER
UNIVERSITY OF WOLLONGONG
DEPARTMENT OF COMPUTING SCIENCE
PO BOX 1144
WOLLONGONG NSW 2500.
AUSTRALIA.

JAN 02 1990 Sue DeAngelis

The University of Wollongong
NORTHFIELDS AVE
WOLLONGONG
NSW
AUSTRALIA

Date: DEC 15, 198

OVERSEAS REMITTANCE ADVICE
No: 1512023

Supplier: UNIVERSITY OF WATERLOO 409324
DEPT OF COMPUTER SCIENCE
WATERLOO ONTARIO
CANADA N2L 3G1

Currency: $ US

INVOICE DATE	OUR REF	ORDER	INVOICE	DETAILS		AMOUNT
11 DEC 89	015466	086562	C/W/O	RESEARCH REPORTS		4.00
				TOTAL:	US	4.00
				CONVERTED AT .7809 =	$A	5.12
				BANK FEE		2.00
				REMITTANCE TOTAL:	$A	7.12

Correspondence regarding this payment should be addressed to The Finance Officer at the above address. Telex enquiries may be made to AUST 29022.

Please quote this advice number on any correspondence.

LUND UNIVERSITY
Department of Computer Sciences

Date
89-11-15

Dept. of Computer Science
University of Waterloo

Dear Sir,

Could you please send me a copy of the following Research Report:

CS-89-07 - Delivering a better introductory course in computer science,
A. Dyck, J. Black, K. Booth

Thank you,

[illegible]

Rolf Karlsson

Sent Dec. 4/89

Mailing address
Box 118
S-221 00 LUND
Sweden

Office address
Ole Römers väg 3
LUND
Sweden

Telephone
Nat: 046 - 10 80 30
Int: + 46 - 46 10 80 30

Telex
33533
LUNIVER S

JOSÉ CUNHA
APARTADO 4276
1507 LISBOA CODEX
PORTUGAL

Research Report Secretary
Dept. of Computer Science
University of Waterloo
Waterloo
Ontario N2L 3G1
Canada

November 21, 1989

Dear Sirs,

Enclosed please find a check for $2.00, regarding a request for sending me the following report from your department:

Research Report CS-89-07
'Delivering a better introductory course in Computer Science'
authors: A. Dyck, J. Black, K. Booth.

Thanks for your attention.

Yours Sincerely,

José C. Cunha

rec'd cheque & sent report Nov. 30/89

NOV 13 1989

UNIVERSITÄT ULM

Abteilung Theoretische Informatik

Universität Ulm, Oberer Eselsberg, D-7900 Ulm

Leiter: Prof. Dr. Uwe Schöning

University of Waterloo
Dep. of Computer Science
Waterloo, Ontario
CANADA N2L 3G1

Oberer Eselsberg
Postfach 4066
D-7900 Ulm/Donau
Telefon: 0731/176-3872
Telefax: 0731/176-2038

November 9, 1989

Dear ladies and gentlemen:

Please send us the following paper:

- CS-89-07—Delivering a better Introductory Course in Computer Science by Arnie Dyck, Jay Black, Kelly Booth

We thank you very much.

Sincerely

I.A. [illegible]

Prof. Dr. Uwe Schöning

sent Nov. 20/89

X If you would like to order individual reports please forward your order, along with a cheque or international bank draft payable to the **Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, to the Research Report Secretary.**

If you wish to receive all reports published in 1989 our **Subscription Rate** is **$125.00** Canadian. Please refer to the above paragraph for payment instructions.

Please indicate your current mailing address.

MAILING ADDRESS:

Dept. of Computer Science Business Office
A.V. Williams Bldg. #115, Room 4172
University of Maryland
College Park, MD 20742 U.S.A.
Attn: Karen White

Rec'd cheque
Sent
89-07
Sept. 5/89

X If you would like to order individual reports please forward your order, along with a cheque or international bank draft payable to the **Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, to the Research Report Secretary.**

If you wish to receive all reports published in 1989 our **Subscription Rate** is $125.00 Canadian. Please refer to the above paragraph for payment instructions.

Please indicate your current mailing address.

MAILING ADDRESS:

Dept. of Computer Science Business Office
A.V. Williams Bldg. #115, Room 4172
University of Maryland
College Park, MD 20742 U.S.A.
Attn: Karen White

Rec'd cheque
Sent
89-07
Sept. 5/89

rder

CS-89-07 - DELIVERING A BETTER INTRODUCTORY COURSE IN COMPUTER SCIENCE

AUTHORS: Arnie Dyck, Jay Black, Kelly Booth

ABSTRACT:

In revising the introductory computer science curriculum at the University of Waterloo, a number of general issues arose. The first was the need to integrate many fundamental topics of computer science into a small number of courses, to provide an overview of the field for students a variety of potential majors. The second was the requirement that the time spent by students (especially programming time) be in proportion to the weight of the course within the overall curriculum. Finally, we wished to close the perceived gap between "computer literacy" courses (where experience with state-of-the-art desktop applications predominates) and more traditional computer science courses (where the emphasis is on system design and implementation). The first issue is reflected in the choice of a "spiral" approach that covers theoretical and practical topics in computer science. The elimination of "open-ended" assignments in favour of time-limited laboratory sessions with preparatory tutorials complements the lecture presentations both with hands-on experience using commercial application packages and with programming assignments where students implement and test their own algorithms. In designing and implementing the new curriculum it became clear that the logistics of delivering the course material were at least as important to success as were the choice of the material itself. A support team of faculty, staff, graduate student teaching assistants, and undergraduate tutors provides a diverse skill set that has proven to be a cost-effective delivery system for the new curriculum.

PRICE: $2.00

CS-89-08 - DESIGNING AND MODELING VLSI SYSTEMS AT THE

REGISTER-TRANSFER LEVEL

AUTHOR: Farhad Mavaddat

ABSTRACT:

In this paper we discuss some of the properties that a design abstraction must have, and make suitable proposals for the RT abstraction. To this effect, we propose a small set of design primitives with well-defined behaviours, and give rules for their composition, all within the framework of a strongly-typed signal environment known as the SDC Model of register-transfer design. The primitives are chosen to permit a mathematical treatment of designs, but they are also highly design-oriented, and provide the designer with a useful and friendly set of building blocks. We also present user-friendly graphical and textual interfaces to the model. To prove our designs correct, we use Algorithmic State Machines (ASM) as the frameworks for analyzing and reasoning about an SDC-based design.

PRICE: $2.00

PURCHASE ORDER

UNIVERSITY OF MARYLAND

COLLEGE PARK, MARYLAND

20742-5115

SHOW THIS ORDER NUMBER ON ALL SHIPMENTS, CORRESPONDENCE AND INVOICES

S- 274292 -P

REFERENCE BID NO. OR CONTRACT NO.	AMOUNT

REQ. NO.	FAS NO.	FISCAL YR.
70	01-1-12620-4316	88/89

TO
Dept. of Computer Science
University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1
AttN: Research Report Secretary

SHIP TO
University of Maryland PO#S274292P
Dept. of Computer Science
Business Office
BLDG. NO. 115 ROOM NO. 4172
College Park, Maryland 20742
ATTN: Karen White

Subject to purchase terms & conditions, furnish goods and/or services shown below.

DATE	DELIVER ON OR BEFORE	TERMS:	F.O.B.
8/17/89	10/5/89	Net	Destination

ITEM NO.	DESCRIPTION OF ARTICLES OR SERVICES	QUANTITY	UNIT OF QUANT.	TOTAL ESTIMATED COST	UNIT PRICE	TOTAL ACTUAL COST	SUBCODE
1.	CS-89-07 - Delivering a Better Introductory Course in Computer Science AUTHORS: Arnie Dyck, Jay Black, Kelly Booth	1	ea.	1.73		~~$2.00~~ Canadian Dollars	

PAID BY COLLEGE PARK WORKING FUND

22788

CHECK NO.________________ATTACHED.

ROUTING INSTRUCTIONS
VENDOR SHIP VIA

QUESTIONS CONCERNING THIS ORDER SHOULD BE REFERRED TO THE BUYER: PERRY/at 6743

Canadian

PURCHASE TERMS AND CONDITIONS

NOTE: Terms & conditions continued on reverse side.

1. A separate invoice in TRIPLICATE for this purchase order or for each shipment thereon shall be rendered immediately following shipment. All copies of invoices must be forwarded directly to the Accounts Payable Department, South Administration Building, University of Maryland, College Park, Md. 20742.
2. The vendor's/contractor's Federal identification number or social security number must be included on the invoice.
3. This purchase order number must be shown on all related invoices, delivery memoranda, bills of lading, packages, and/or correspondence.

FAILURE TO COMPLY WITH THESE TERMS WILL RESULT IN THE INVOICE BEING RETURNED TO YOU.

UNIVERSITY OF MARYLAND
By Monique C. Perry 8/21/89
AUTHORIZED SIGNATURE

NOTE: THE UNIVERSITY OF MARYLAND IS EXEMPT FROM THE FOLLOWING TAXES:
1. State of Maryland Sales Tax by Certificate No. 30002563
2. District of Columbia Sales Tax by Exemption No. 9199 79411 01
3. Manufacturer's Federal Excise Tax Registration No. 52 730123K.

UNIVERSITÄT KARLSRUHE
FAKULTÄT FÜR INFORMATIK
Institut für Algorithmen und Kognitive Systeme

Prof. Dr. J. Calmet

Technologie-Fabrik Karlsruhe
Haid-und-Neu-Straße 7
D-7500 Karlsruhe 1
Tel. 07 21/608-4208
02.08.1989

Department of Computer Science
University of Waterloo

Waterloo, Ontario N2L 3G1
Canada

Ladies and Gentlemen,

I would like to order the following reports:

CS-89-07 "Delivering a better Introduction Course in Computer Science"
Arnie Dyck, Jay Black, Kelly Booth

CS-89-13 "Solving Systems of Algebraic Equations, or the Interface between Software and Mathematics"
Gaston H. Gonnet, Michael B. Monagan

For administrative reasons we are not able to include a check in this order. Please send us a bill, then our University will send the money as soon as possible.
Please send the reports to

Prof.Dr. Jacques Calmet
Universität Karlsruhe
Institut für Algorithmen
und Kognitive Systeme
Haid-und-Neu-Str.7
D-7500 Karlsruhe, FRG

sent Aug. 25

Thank you in advance.

[illegible]

(U. Beelitz, secretary)

TO OPEN SLIT HERE FIRST

AEROGRAMME

BY AIR MAIL • PAR AVION

Fix Current Aerogramme Postage Stamp

The Secretary
Department of Computer Science
University of Waterloo
WATERLOO
ONTARIO N2L 3G1
C A N A D A

(COUNTRY OF DESTINATION)

Approved by the Australia Post for acceptance as Aerogramme No. 58

SECOND FOLD HERE

SECOND FOLD HERE

SENDERS NAME AND ADDRESS

D C Edwards (Mrs)
Dept of Electrical Engineering
& Computer Science
University of Newcastle
New South Wales
AUSTRALIA 2308

POSTCODE

FIRST FOLD HERE

FIRST FOLD HERE

THE UNIVERSITY OF NEWCASTLE

NEW SOUTH WALES, 2308

AUSTRALIA.

DEPARTMENT OF ELECTRICAL ENGINEERING

and

COMPUTER SCIENCE

14 August 1989

The Secretary
Department of Computer Science
University of Waterloo
Waterloo
ONTARIO N2L 3G1

Dear Sir/Madam,

RESEARCH REPORTS 1989 (January to April)

Would you kindly let me have one copy of the following report please:

CS-89-07 Delivering a Better Introductory Course in Computer Science

Authors: Arnie Dyck, Jay Black, Kelly Booth.

Thanks.
Yours sincerely,

sent Aug. 25

[illegible]

Diana C. Edwards (Mrs)
Secretary
Computer Science

DEPARTMENT OF COMPUTING
AND INFORMATION SCIENCE
Phone # (613) 545-6050
FAX # (613) 545-6513

Queen's University
Kingston, Canada
K7L 3N6

July 27, 1989

Department of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 3G1

Attn: Research Report Secretary

Dear Secretary:

Could you please forward to me a copy of CS-89-07 "Delivering a Better Introductory Course in Computer Science", by Arnie Dyck, Jay Black, Kelly Booth. I understand the cost for this report is $2.00. If you could invoice me for this amount I will send a cheque to you. Thank you.

Sincerely

Henk Meijer

Henk Meijer
Associate Professor

HM/hb

sent
Aug. 22

yne State University
llege of Liberal Arts

Department of
Computer Science
Detroit, Michigan 48202
(313) 577-2476
(313) 577-2477

August 7, 1989

Department of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 3G1, Canada

Dear Friends:

Enclosed is $4.00 for copies of the following technical reports:

1. CS-89-07 Delivering a Better Introductory Course in Computer Science

2. CS-89-14 Reasoning about Functional Dependencies Generalized for Semantic Data Models

rec'd payment
and sent reports
Aug. 17

Sincerely,

[illegible]

William I. Grosky
Professor

CS-89-13 - SOLVING SYSTEMS OF ALGEBRAIC EQUATIONS,

OR THE INTERFACE BETWEEN SOFTWARE AND MATHEMATICS

AUTHORS: Gaston H. Gonnet, Michael B. Monagan

ABSTRACT:

This paper describes two fundamental aspects of the solution of systems of algebraic equations: the use of *substitution* and the use of a *complexity function* to determine, at each step, which of several methods /equations /unknowns to use. These techniques, which have received little attention in the past, can be viewed as in the interface between mathematics and software. An implementation of these techniques in *Maple* [Cha 83] is described. Timing results comparing the Maple, *Macsyma* [Mos 74] and *Reduce* [Hea 71] symbolic algebra systems on a range of sample problems are presented. The proposed method proves to be superior to other techniques.

PRICE: $2.00

CS-89-14 - REASONING ABOUT FUNCTIONAL DEPENDENCIES

GENERALIZED FOR SEMANTIC DATA MODELS

AUTHOR: Grant E. Weddell

ABSTRACT:

We propose a more general form of functional dependency for semantic data models that derives from their common feature in which the separate notions of *domain* and *relation* in the relational model are combined into a single notion of *class*. The feature manifests itself in a richer terminological component for their query languages in which a single term may traverse any number of properties (including none). We prove the richer expressiveness of this more general functional dependency, and exhibit a sound and complete set of inference axioms. Decision procedures are developed that apply when the dependencies included in a schema correspond to keys, or when the schema itself is acyclic. The theory is then extended to include a generalization of select-join queries. Of particular significance is that the queries become an additional source of functional dependency constraints. Finally, we outline several applications of the theory to various problems in physical design and in query optimization. The applications derive from an ability to predict when queries can have at most one solution.

PRICE: $2.00

If you would like to order individual reports please forward your order, along with a cheque or international bank draft payable to the **Department of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, to the Research Report Secretary.**

If you wish to receive all reports published in 1989 our **Subscription Rate** is $125.00 Canadian. Please refer to the above paragraph for payment instructions.

Please indicate your current mailing address.

MAILING ADDRESS: Prof. Louis Becker
2427 Central Road
Glenview, IL. 60025
U.S.A.

Please send report no CS-89-07.
Thank you.

rec'd payment
+
sent
Aug. 17

IDENTIFICATION: UNIVERSITY OF WATERLOO

CHEQUE NUMBER: 234022

REQ DESCRIPTION INV	AMOUNT	DESCRIPTION	AMOUNT
550190	2.00	Other Payment CS8907	
Att'n M WILSON COMP SCI	538 HP		

CARLETON UNIVERSITY OTTAWA, CANADA K1S 5B6

Carleton University
Ottawa, Canada K1S 5B6

27 July 1989

Research Report Secretary
Department of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 3G1

rec'd payment + sent Aug. 17

Enclosed please find a cheque in the order of $2.00 to cover the cost of the following report:

CS-89-07 Delivering a better introductory course in Computer Science

Authors: Arnie Dyck, Jay Black, Kelly Booth

[illegible] for
J. Pugh
Director
School of Computer Science

University of Regina

Department of Computer Science Regina, Saskatchewan Canada S4S 0A2 (306) 585-4633

August 1, 1989

rec'd payment
sent Aug. 17

Technical Report Secretary
Department of Computer Science
University of Waterloo
Waterloo, Ontario
N2L 3G1

Dear Sir:

I would like to order technical report CS-89-07 - Delivering a Better Introductory Course in Computer Science. Please find enclosed $2.00 cash in payment for the report.

Thank you.

Sincerely,

R. B. Maguire/mc

R. B. Maguire
Head, Computer Science

RBM/mc

Southern Alberta
Institute of Technology
1301 - 16 Avenue N.W.
Calgary, AB. Canada T2M 0L4

MATHEMATICS, PHYSICS AND COMPUTING DEPARTMENT
Phone: 284-8543 (403)

August 3, 1989

Department of Computer Science
The University of Waterloo
WATERLOO, Ontario
N2L 3G1

Attention: Research Report Secretary

RE: CS-89-07 - DELIVERING A BETTER INTRODUCTORY COURSE IN COMPUTER SCIENCE

Please forward the above referenced document by Authors Arnie Dyck, Jay Bloack, and Kelly Booth.

You ~~will~~ find enclosed a Postal Money Order in the amount of $2.00 as payment for same.

Thank you.

Yours truly,

[illegible]

rec'd money and sent report Aug. 17

H. K. Myhre
Department Head
Mathematics, Physics, and Compunting

HKM/meh

Enclosure

Offering courses, seminars and other consultative services in mathematics, physics, computer sciences, pre-technology upgrading, geophysics, engineering science technology, computer managed learning, and general studies.

Printing Requisition/Graphic Services

54242

1. Please complete unshaded areas on form as applicable.
2. Distribute copies as follows: White and Yellow to Graphic Services. Retain Pink Copies for your records.
3. On completion of order the Yellow copy will be returned with the printed material.
4. Please direct enquiries, quoting requisition number and account number, to extension 3451.

TITLE OR DESCRIPTION: Delivering A Better Introductory Course In Computer Science CS-89-08

DATE REQUISITIONED: March 3/89 | DATE REQUIRED: ASAP | ACCOUNT NO.: 1 2 6 4 4 3 1 0 7

REQUISITIONER- PRINT: A. Dyck | PHONE: 3475 | SIGNING AUTHORITY: Ursula Thoene

MAILING INFO – NAME: Sue DeAngelis | DEPT.: C.S. | BLDG. & ROOM NO.: DC 2314 | [✓ DELIVER] PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES: 14 | NUMBER OF COPIES: 100

TYPE OF PAPER STOCK: BOND NCR ___ PT. COVER BRISTOL SUPPLIED Alpac Ivory ~~140M~~

PAPER SIZE: 8½ x 11 8½ x 14 11 x 17 10x14 Glosscoat 10 pt Rolland Tint

PAPER COLOUR: WHITE [✓] ______ | INK: [✓ BLACK] ______

PRINTING: 1 SIDE ___ PGS. [✓ 2 SIDES] ___ PGS. | NUMBERING FROM ___ TO ___

BINDING/FINISHING: [✓ COLLATING] STAPLING ___ HOLE PUNCHED PLASTIC RING

FOLDING/PADDING: 7x10 saddle stitched | CUTTING SIZE

Special Instructions

* Please photo reduce for Beaver Cover format.

Both cover and inside in black ink please.

COPY CENTRE — OPER. NO. | BLDG. | MACH. NO.

DESIGN & PASTE-UP — OPER. NO. | TIME | LABOUR CODE: D01, D01, D01

TYPESETTING — QUANTITY: PAP000000 T01; PAP000000 T01; PAP000000 T01

PROOF: PRF; PRF; PRF

NEGATIVES — QUANTITY | OPER. NO. | TIME | LABOUR CODE: FLM C01 (×5)

PMT: PMT C01 (×3)

PLATES: PLT P01 (×3)

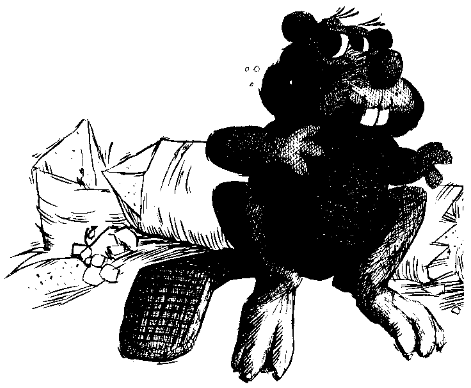
STOCK: 001 (×4)

BINDERY: RNG B01; RNG B01; RNG B01; MIS000000 B01

OUTSIDE SERVICES

$ ______ COST

TAXES – PROVINCIAL FEDERAL | GRAPHIC SERV. OCT. 85 482-2



Delivering A Better Introductory Course In Computer Science

V.A. Dyck
J.P. Black
K.S. Booth

Research Report
CS-89-07

February, 1989

DELIVERING A BETTER INTRODUCTORY COURSE IN COMPUTER SCIENCE

V.A. Dyck, J.P. Black, and K.S.Booth
Department of Computer Science
University of Waterloo

Abstract

In the process of revising the first and second year computer science curriculum at the University of Waterloo, a number of issues arose that should be of general concern. The first was the need to integrate many of the fundamental topic areas of computer science into a small number of introductory courses, to provide an overview of the field both to students who intend to major in computer science and to those for whom the introductory courses will be the only formal training in computer science. The second was the requirement that the time spent by students (especially the time spent programming) be in proportion to the weight of the course within the overall curriculum. Finally, we wished to close the perceived gap between "computer literacy" courses (where experience with state-of-the-art desktop applications predominates) and more traditional computer science courses (where the emphasis is on system design and implementation). The first issue is reflected in the choice of a "spiral" approach that covers theoretical and practical topics in computer science during the first two years (four term courses). The elimination of "open-ended" assignments in favour of time-limited laboratory sessions with preparatory tutorials complements the lecture presentations both with hands-on experience using commercial application packages and with programming assignments where students implement and test their own algorithms. In designing and implementing the new curriculum it became clear that the logistics of delivering the course material were at least as important to success as were the choice of the material itself. A support team of faculty, staff, graduate student teaching assistants, and undergraduate tutors provides a diverse skill set that has proven to be a cost-effective delivery system for the new curriculum.

1. Introduction

The Faculty of Mathematics and the Department of Computer Science at the University of Waterloo recently completed a major three-year redesign of our first and second year courses in Computer Science. Rarely does such an opportunity arise to completely rethink the course curriculum and delivery for such an extensive set of courses.

The Faculty of Mathematics at the University of Waterloo has one of the largest classes of mathematics students in the world. Each year it admits approximately 950 students to a common first-year program. Students choose an area of specialization in second year. Typically, students divide equally among majors in computer science, majors in mathematics with accounting or business, and majors in traditional areas of mathematics (for example, applied, pure, statistics, combinatorics or optimization). All three groups are expected to have varying levels of computer literacy upon graduation. Some complete only the two first-year courses, some only the four courses in the first two years, and some go on to major or minor in computer science.

We deliberately adopted a "spiral" approach in each of the two years: each year was in some sense to include a complete view of all important areas of computer science, so that students who only completed one year before continuing in a non-CS program would have been exposed to most of the theoretical and practical foundations we considered to be important. A secondary consideration was that many students are in our co-operative education program, and so must be prepared for a four-month work term in industry after only one four-month term on campus.

For reasons explained later in this paper, we felt that a significant reorganization of the curriculum was essential. A major consideration in the development of the new courses was the effective delivery of the material, both in its theoretical aspects and its practical applications. Considerable thought was given to the subject material, organization and effectiveness of the delivery of the new courses. We believe we have designed courses which are more academically satisfying, more modern in content and approach, and which combine effective delivery of material to students with efficient use of limited human and physical resources.

Thus we had the following goals in revising the first two courses:

- to modernize the curriculum to include an overview of the major theoretical and practical aspects of computer science,
- to expose the students to modern microprocessor hardware and software, and
- to control the amount of time spent by the students on these courses.

This paper discusses the redesign of the course sequence and the implementation of the first two courses in it. In Section 2, we describe our goals for the new curriculum, and some of the environmental pressures and constraints which influenced the final form of the courses. We also include details on the particular goals of the first two courses. Section 3 discusses various aspects of the course implementation, including the relationships between lectures, tutorials, and laboratories, problems of scale and resources, and the hardware and software environment. Section 4 presents a summary and some conclusions we have drawn from our experience to date.

2. Goals of the New Computer Science Curriculum

Historically, the first two courses in computing emphasized the development of problem-solving and programming skills. A major goal was to have the students programming the computer as quickly as possible. A mainframe environment was used as the course delivery system. After an hour or two of instruction on the system, students were largely left to their own resources to solve the two to three programming questions each week. Students typically spent ten to fifteen hours per week on completing their assignments, often debugging their programs by "friction." (Students seemed to think that if the lines of code were shuffled often enough a correct solution magically might be found by "wearing down" the bugs.) More importantly, students had relatively poor debugging tools, and no training in their use. Instructors in other courses resented the excessive time students spent on computer science, and the computer science instructors resented the students' preoccupation with programming.

In addition to these deficiencies, students rarely gained a perspective on the hardware and software they used, nor were they introduced to the wide variety of modern application software. Not surprisingly, they tended to see the computer primarily as a device for programming. Even though increasing numbers of freshmen had been introduced to computer concepts in high school, very rarely did their background include instruction in use of this new applications software.

While we did not explicitly adopt the ACM curriculum for CS 1 '84 [1] and CS 2 '84 [2], a comparison is useful. These two model courses cover the fundamentals of programming in a high-level language, especially procedural abstraction in CS 1 '84, and data abstraction and abstract data types in CS 2 '84. Our courses follow this same general approach, but we wanted to meet some other specific needs: we wanted our students to have some early exposure to numerical computing, especially for those continuing in mathematics but not CS; we wanted to include a fairly broad range of theoretical topics at an introductory level; and we wanted to give students some exposure to modern microprocessor programming and application environments, such as they were likely to encounter on their first work terms. Thus, while there are some very strong similarities between our courses and the ACM model courses, our curriculum gives students a much broader introduction to both the theoretical and practical sides of computer science.

Our revised first course begins with an overview of hardware and how the components function together. This is followed by an overview of software, from the operating or system level through various levels of languages to application software. A major portion of the first course continues to be devoted to the development of algorithmic solutions to problems and a study of high-level, block-structured language features to solve these problems (currently, we teach a dialect of Pascal). Students are taught problem solving and programming with emphasis on proper design, correctness and efficiency of programs. Both the theoretical underpinnings and the practical implementation are addressed in the course delivery. In the final third of the course, we also touch on topics such as sorting and searching, program verification, algorithmic complexity, relational data bases, and the data-directed design approach to file processing applications. The more theoretical lectures are complemented by laboratory experience, including, in particular, the personal productivity software mentioned above.

In the second course, we take a more detailed look at programming systems including machine language and translation (compilers and interpreters), as well as data structures and storage allocation. The students are also taught a second programming language (FORTRAN 77), which is used for an introduction to numerical computation, including zero-finding, area-finding, linear equations, and simulation. The course concludes with an introduction to undecidability to instill an appreciation of the role of mathematical theory in computer science. The conscious decision to include these advanced topics (algorithmic complexity, program verification, error analysis, undecidability) was taken to emphasize the close relationship between theory and practice.

3. Implementation of the Courses

In addition to the concern for curriculum issues, considerable thought has been given to the delivery of these courses, especially the first, since this is the introduction to computers for many students. There are three hours of lectures per week, devoted to presenting the principles and theoretical background of the material. These are complemented by a one-hour tutorial in small groups which bridges between the lectures and the weekly three-hour lab. For example, when learning a new programming language, the principles of a language are addressed in lectures, the syntax details are described in tutorials, and reinforcement and practice are provided in the lab.

To address the implementation difficulties of previous courses, the decision was made to emphasize quality rather than quantity of hands-on experience. Adopting the model used in the more traditional sciences such as biology, chemistry and physics, students are expected to attend a scheduled and structured weekly laboratory session. In these sessions they are exposed to the complementary, practical side of the course. They learn how to use a variety of application software as well as the editing and debugging tools necessary to program effectively.

Each week students are given a 10-12 page document describing the current lab. The tutorial provides some necessary background material to prepare for the lab. Students are expected to read the lab material carefully in advance, and to come to the lab prepared to proceed through the specified instructions and exercises. During the labs, trained graduate teaching assistants and full-time undergraduate tutors provide the necessary guidance. At the end of the three-hour lab, students submit their materials for consideration and evaluation by the tutors.

This structure allowed the course development team to address a number of the difficulties with earlier courses. Through a structured lab environment, students are encouraged to make efficient use of their time on the computer. Preparation must be completed in advance or it becomes impossible to complete lab exercises in the prescribed amount of time. By providing trained support staff, students are encouraged to explore and use various software tools effectively as they solve a variety of lab exercises.

During the first few weeks when the lecture schedule provides hardware and software material on which future programming labs will be based, students are introduced to the practical use of a computer and to software such as word processors and spreadsheets. Thus, immediate hands-on experience is given in the labs prior to students knowing much about how computers really work. Once a foundation has been laid in lectures, labs dealing with programming are introduced. By this time students have already used the computer system for several weeks and can concentrate on the programming environment. Later in the course, students are taught to manipulate quantities of data in both a high-level language and using a modern database management system as lectures, tutorials and lab sessions proceed more in parallel.

From a more general point of view, material is first introduced in lectures, and students have an opportunity to practice what has been taught the following week. Tutorials are typically held two working days before the labs, giving some time for lecture material to be digested before the tutorial, and for the tutorial material to be used by the students to prepare for the labs.

Corresponding to these three types of student contact are three types of teaching staff: faculty members for the lectures, graduate student tutorial assistants (TAs) for the tutorials, and full-time undergraduate tutors for hands-on assistance during the labs. However, TAs are also present during labs, and we have found that the combination of their more mature approach and the less experienced approach of the tutors has worked exceptionally well in practice.

By the end of the first course, students have been introduced to a variety of application software, they have a working knowledge of a high-level programming language, and they have a firm theoretical foundation underlying the practical experience.

From the beginning, it was necessary to face the realities of delivering a course to 950 students at one time. Consequently, it was decided to select a group of 60 volunteers to participate in an initial, prototype offering of the courses. Since we were unable to find a textbook which completely fulfilled our needs, instructors prepared a set of lecture notes to supplement an introductory textbook on Pascal programming.

For the full-scale offering of the course, students were divided into 4 lecture sections. For the tutorials and the labs, students were grouped into 16 sections of 60 students each. The organization for the tutorials and labs was handled by a faculty member serving as lab instructor. A staff of 8 TAs (working 10-12 hours per week) and 4 undergraduate co-op student tutors (working 35 hours/week) were assembled to provide support. In addition, a full-time head tutor was hired to coordinate the computer hardware and software support and to act as a lab trouble-shooter.

Each week, the lab instructor presented a "model" tutorial to one lab section, attended by all of the TAs. They were provided with a set of tutorial notes and each presented tutorials to two sections of students based on the model given by the lab instructor. The TA was paired with two undergraduate tutors to provide support during the corresponding lab sessions for the same two sections of students. Materials submitted at the end of lab were marked by the tutors and returned to students at the next tutorial.

To co-ordinate the weekly activities, two regular meetings were scheduled. The lecturers and lab instructor met to discuss the curriculum issues and the design of weekly materials for lectures, tutorials and labs. The second meeting (immediately afterwards) involved all staff and provided the necessary coordination of the various course activities. Both TAs and tutors were given sufficient opportunity to comment on the previous week's activities, including an evaluation of the tutorial and lab sessions. The meetings were highly successful in developing a sense of team spirit on the part of all participants and in determining the areas of the course presentation that required refinement for future weeks and offerings.

Because of the number of students and lab sections, we had to face problems of scale in scheduling lab resources and people, and in collecting and grading assignments and examinations. We were able to collect assignments automatically from the student's computer accounts, which greatly simplified the problem of collecting and collating several thousand pieces of paper per week (more than 900 students times multiple questions per lab). We spent considerable time designing the midterm and final examinations so they could be marked in a reasonable amount of time and be marked consistently by a team of more than fifteen people.

However, it is important to emphasize that not all of what we did is applicable only to large classes. Our lecture/tutorial/lab delivery based on tutorial and lab sections of about 60 students was intended to address explicitly the problem of providing lower student/staff ratios for at least part of the course. We found it a significant advantage that the teaching assistant who conducted a tutorial for a particular group of students was also present during the lab for that group.

The prototype and first full-scale versions of the course were offered on a mainframe environment. The types of software that could be taught effectively were severely limited by such a system. Furthermore, it was painful to watch students struggle to cope with the vagaries of the system.

For example, the command structure in each part of the system varied and required the student to repeat the learning process over and over again. It was clear that a microcomputer environment with consistent, user-friendly software was essential to allow students to focus on the content of the course rather than on the system used to present the material. During 1988, the lab materials were converted and enhanced based on a lab of modern microcomputers.

Because of favorable experiences with the Apple Macintosh in supporting other courses at the University of Waterloo, the course development team was enthusiastic about a potential Macintosh solution. The Macintosh II was chosen as the workstation for delivering this course. It provided a number of advantages: the uniform, easy-to-use interface, a variety of application software, a suitable programming environment and greater potential for future expandability. Application software selected for the course includes: word processing, spreadsheets, graphics, and database management. Students are also introduced to a highly interactive Pascal programming environment including a symbolic and a hexadecimal debugger.

Managing the physical and human resources for such a large lab presents some unique challenges. For example, the need to provide software, data files, and peripherals for 950 students makes the provision of some sort of distributed system a necessity. These requirements are easily accommodated on a mainframe system, but in moving to a microcomputer environment, we needed some other way to meet these needs.

The University of Waterloo has developed a number of educational networks, including MacJANET for the Apple Macintosh. In this lab, 60 Macintosh II workstations are linked to three interconnected Macintosh II network servers. The servers store and control the use of course software and data files, as well as all the student files. As a result, problems of managing program and data disks are minimized, and software copyright restrictions can be enforced.

We still hope to acquire one more suite of software for electronic mail. A lab environment isolated from the rest of the campus internet is no longer a convenient alternative. We feel that our students should be exposed to and have regular access to this increasingly important communication medium, beginning with registration in first year and continuing throughout their undergraduate career. As of this writing, a prototype mail system for MacJANET is being installed. Modifications to fully support automatic collection of assignments are being considered.

5. Summary and Conclusions

We have completed two full-scale offerings of each of the first two courses in our sequence. Our curriculum effort is now directed more towards the second-year pair of courses. The first of these has had one full-scale offering. The second has had only a prototype offering. We will shortly be facing less drastic revisions to our third-year curriculum, mostly due to the changes to the first two years. As for fourth year, where individual instructors teach a larger number of specialized courses to much smaller classes, we currently expect there will be little effect beyond the normal modifications due to the evolution of the state of the art.

We do not yet have enough experience with the entire four-course package to evaluate this significant upheaval in our curriculum. This will come over the next year or so as we get more experience in the second-year courses, and are able to observe the effects on third- and fourth-year students of our non-standard initiation to computer science. In the interim, we are able to draw some specific conclusions about the two first-year courses.

First, curriculum development on this scale is tremendously expensive from all points of view: faculty time to develop new course materials (lectures, lecture notes, tutorials, lab exercises and assignments), faculty time to hire and manage between ten and twenty course personnel per term, salary costs for tutorial assistants, co-op tutors, and the full-time head tutor, and of course, hardware and software costs for the lab itself. The fixed-time lab period, in particular, has enabled us to schedule our hardware resources and the course personnel very effectively. Similarly, the relatively expensive use of graduate student teaching assistants has been balanced by the use of more cost-effective full-time undergraduate co-op students for marking and lab supervision.

The overall costs are justified by what we feel have been our successes.

We are quite pleased at the new orientation of our freshman class: they certainly have had exposure to a much larger number of topics and ideas than was the case in our earlier offerings. Informal feedback from their employers during the first co-op work term has been very positive, although some employers needed to rethink job responsibilities to make good use of a different skill set that, in particular, included fewer programming skills than before. However, this was more than balanced by skills in such areas as word processing, spreadsheets, data management, and debugging, and by a somewhat higher degree of theoretical understanding.

We feel that the lecture/tutorial/lab format and scheduling has worked extremely well for both staff and students, although we seem currently to be having a problem with lecture attendance. This is perhaps not uncommon in such large lecture sections. The use of three staff in each lab period has gone a long way towards making the lab environment a pleasant and productive one for the students.

An extensive student questionnaire has confirmed that student workload is now in line with other courses, and that the computer science courses are perceived to be no more difficult than the algebra and calculus courses required of all our freshmen. No new major problem areas were revealed by questionnaire results.

6. Acknowledgements

The prototype course was developed by K.S. Booth and V.A. Dyck. Subsequent offerings of the course were enhanced by the efforts of J.P. Black. The course team greatly appreciates the efforts of J.W. Wong and J.W. Graham, S.L. Fenton, and C.M. Durance in acquiring and installing the lab hardware and software. The authors wish to acknowledge the generous support of Apple Canada Incorporated and the Computer Systems Group at the University of Waterloo.

References

[1] E. B. Koffman, P. L. Miller, and C. E. Wardle, "Recommended Curriculum for CS1, 1984," *Comm. of the ACM*, **27**(10), pp. 998–1001, Oct. 1984.

[2] E. B. Koffman, D. Stemple and C. E. Wardle, "Recommended Curriculum for CS2, 1984," *Comm. of the ACM*, **28**(8), pp. 815-818, Aug. 1985.

V.A. Dyck is an associate professor of Computer Science and currently holds the position of associate dean for undergraduate studies in the Faculty of Mathematics at the University of Waterloo. J.P. Black is an associate professor of Computer Science and director of the Shoshin Lab. K.S. Booth is a professor of Computer Science and is director of the Institute for Computer Research.

CS 131

Principles of Computer Science I

Objectives
To introduce students to the use of computers in problem solving.

Intended Audience
CS 131 is a required course for all programs in the Faculty of Mathematics. Math students normally complete CS 131 in their 1A term.

Prerequisites/Corequisites/Possible Successors
Prerequisites: Grade 13 Mathematics
Successors: CS 132

References
There is no current course textbook.

Schedule
2 hours of lecture, 1 hour of tutorial and a compulsory 3 hour laboratory per week.

Terms Available
Fall and Winter

Notes
1. Latest revision: August, 1986

Outline
This is the first in a series of four courses designed to provide an introduction to the principles of computer science. This course concentrates on the fundamental notions of algorithmic solutions to problems and issues related to the implementation of algorithms as programs in high-level languages.
Overview of Computer Science (4 hours)
Computers in historical perspective. Components of a computer. Using a computer. Introduction to general computer problem solving. Programming languages from low-level to high-level. The software development process. The role of theoretical tools in computer science. Examples of application and system software in practice.
Introduction to Algorithms (3 hours)
Formally specifying procedures. Constants, variables and scalar data types, input/output. Control structures: conditional, iteration, and selection. Examples of algorithms to solve problems of interest.
Structured Programming (2 hours)
Top-down design, step-wise refinement. Documentation. Modularity, procedures and functions, and parameter passing. Arrays as structured data types.
Building Correct Algorithms (3 hours)
Systematic testing, debugging, run-time profiling. Introduction to formal program verification techniques. Method of invariant assertions, partial correctness, termination. Examples based on searching and sorting problems.
Building Efficient Algorithms (3 hours)
Measures of algorithmic complexity. Upper and lower bounds on complexity. Asymptotic analysis of algorithms, "Oh" notation. More searching and sorting examples.
Recursion (1 hour)
Notion of a recursive procedure or function. Proving correctness and efficiency of recursive algorithms. Quicksort.
Record and File Management (2 hours)
Records and fields as structured data types. Basics of record management. Files of records. End-of-file handling. Data-directed design of programs.
Introduction to Data Base Management (2 hours)
Logical and physical representation of data. Using secondary storage techniques. Data base management systems.
File Processing Utilities (1 hour)
Auxiliary storage devices (tapes and disks). Concept of a utility program. Examples including media conversion routines and sort utilities.
File Maintenance and Report Generation (1 hour)
Generalized file naming. Additions, deletions, and changes to a file. Generating single and multiple reports.
Communications and Networks (2 hours)
Hardware and software communication protocols. Electronic mail in a network environment. Privacy and security considerations.

CS 132

Principles of Computer Science II

Objectives
To broaden students' understanding of fundamental issues related to problem solving by computer.

Intended Audience
CS 132 is a required course for all programs in the Faculty of Mathematics. Math students normally complete CS 132 in their 1B term.

Prerequisites/Corequisites/Possible Successors
Prerequisites: CS 131
Successors: CS 241

References
There is no current course textbook.

Schedule
2 hours of lecture, 1 hour of tutorial and a compulsory 3 hour laboratory per week.

Terms Available
Fall, Winter and Spring

Notes
1. Latest revision: August, 1986

Outline
This is the second in a series of four courses designed to provide an introduction to the principles of computer science. This course describes the basic architecture of a computer and introduces the notion of machine language programming. The theory of undecidability and the representation of numerical problems are discussed in terms of their fundamental limitations on our ability to solve problems by computer. Techniques for solving numerical problems and the use of data structures and their implementation are introduced.

Overview of Programming Systems (2 hours)
Syntax description using BNF. Pattern matching and regular expressions. Compilers and interpreters. Block diagram of a compiler. Linkers and loaders.

Introduction to Machine Language (2 hours)
The concept of a stored program. Instructions sets and data representation; binary, octal, and hexadecimal number systems.

Linked List Data Structures (3 hours)
Records with pointer fields. Singly- and doubly-linked linear lists, circular lists. Stacks, queues, and deques. Binary search trees.

Hashing Techniques (2 hours)
Open hashing schemes. Handling collisions; probe sequences, overflow buckets

Dynamic Storage Allocation (1 hour)
First-fit and best-fit heuristics. Automatic garbage collection versus explicity deallocation.

Zero-Finding Techniques (2 hours)
Iterative search, bisection, regula falsi, secant method, Newton's method. Conditions for convergence.

Pitfalls of Numerical Computation (1 hour)
Computer representation of numbers; fixed and floating-point systems. Truncation and rounding, roundoff error, cancellation.

Area-Fiding Techniques (2 hours)
Quadrature rules for numerical integration of functions, Riemann sums. Rectangle Rule, Trapezoid Rule, Simpson's Rule. Error estimates.

Solving Linear Equations (3 hours)
Gaussian elimination. Error analysis. Pivoting. Linked list data structures for sparse matrices.

Simulation Techniques (3 hours)
Deterministic simulation. Random number generators, uniform and Gaussian distributions. Probabilistic (Monte Carlo) simulation

Introduction to Decidability (3 hours)
The notion of decidability. The Halting Problem. Undecidable problems related to computer programs.

CS 241

Principles of Computer Science III

Objectives

To describe the relationship between high-level programming languages and the computer architecture that underlies their implementation.

Intended Audience

CS 241 is a required course for some programs in the Faculty of Mathematics and for all programs in the Department of Computer Science. Math students normally complete CS 241 in their 2A term.

Prerequisites/Corequisites/Possible Successors

Prerequisites: CS 132.
Successors: CS 242.

References

There is no current course textbook.

Schedule

2 hours of lecture, 1 hour of tutorial and a compulsory 3 hour laboratory per week.

Terms Available

Fall, Winter and Spring.

Notes

1. Latest revision: August, 1986.

Outline

This is the third in a series of four courses designed to provide an introduction to the principles of computer science. This course covers computer architecture as it relates to the design and implementation of translators for programming languages. A survey of features present in high-level languages and techniques for implementing compilers for such languages is provided.

Basic Machine Architecture (4 hours)
Functional components of a computer; memory, control unit, arithmetic/logic unit, input/output unit. Data representation (review). Machine language; operation codes, addressing modes, indexing, base registers, register designation.

Assemblers (2 hours)
Mnemonic op-codes, pseudo-ops, symbolic constants and addresses, literals.

Programming Language Specification (4 hours)
Syntax vs. semantics. Backus Naur Form, extended BNF, syntax diagrams; equivalent representations of syntax. Syntax-directed translation; parsing, parse trees, expression trees.

The Translation Process (4 hours)
Phases of translation. Interpreters vs. compilers. Lexical analysis and symbol tables. Code generation; Automatic parser generation; shift-reduce parsing, semantic actions. Peephole optimization.

Linkers and Loaders (2 hours)
Resolving external references. Libraries. Linking between different languages.

Block-Structured Languages (2 hours)
Algol-60 inherited scope rule. The run-time stack. Handling recursion. Static vs. dynamic binding.

Parameter Passing Mechanisms (2 hours)
Call by value, call by value-result, call by reference, and call by name. Parameter passing using the run-time stack. Hardware and software features to support parameter passing.

Comparing Programming Languages (4 hours)
Control structures, data structures, and naming structure. Procedural vs. applicative languages.

CS 242

Principles of Computer Science IV

Objectives
To describe the function of modern operating systems and their relationship
to the computer architecture that underlies their implementation.

Intended Audience
CS 242 is a required course for some programs in the
Faculty of Mathematics and all programs in the Department of Computer
Science.
Math students normally complete CS 242 in their 2B term.

Prerequisites/Corequisites/Possible Successors
Prerequisites: CS 241.
Successors: CS 340, CS 350, CS 354, CS 360, CS 369, CS 375.

References
There is no current course textbook.

Schedule
2 hours of lecture, 1 hour of tutorial and a compulsory 3 hour laboratory
per week.

Terms Available
Fall, Winter and Spring.

Notes
1. Latest revision: August, 1986.

Outline
This is the fourth in a series of four courses designed to provide an introduction to the principles of computer science. This course discusses the principles underlying the design and implementation of operating systems, I/O and interrupt handling, and issues related to the use of large distributed systems. Techniques for achieving parallelism, both in hardware and software, are described. Computer techniques are best understood when they are motivated by the need to solve a real problem.
Components of an Operating System (3 hours)
Block diagram of an O/S (review). Peripherals: disks, tapes, terminals, local-area networks.
File Systems (2 hours)
Hierarchical tree-structured file system. Protection mechanisms; access lists and capability lists. Backup and restore procedures.
I/O and Interrupt Handling (2 hours)
Device drivers. DMA and programmed I/O. Priority interrupts: interrupt levels, enabling/disabling interrupts.
CPU Scheduling and Swapping (2 hours)
Multiprogramming vs. multitasking. Context switching.
Memory Management (3 hours)
Memory protection. Virtual memory; paging and segmentation hardware and software. The working set, least-recently-used replacement strategies. The memory cache.
Parallel Processing (4 hours)
Multiprocessors: MIMD and SIMD models of computation. Structuring a program as a collection of parallel tasks. Task synchronization primitives: semaphores, monitors, message passing.
Networking (2 hours)
The RS-232 standard. Telecommunications protocols. Local area networks.
Performance Monitoring (2 hours)
Instrumenting a large software system. Tools for analyzing large software systems.
Modeling and Simulation (2 hours)
Queuing models representing an O/S.
Social Aspects of Computing (2 hours)
Ensuring security and privacy. Health hazards and ethical issues. Copyright and patent law relating to computer hardware and software.