A history of compilers

Peter Sestoft
sestoft@itu.dk

DIKU, University of Copenhagen
2014-02-21

Outline

- What is a compiler?
- A history of the history of ...
- Early autoprogramming systems
- The FORTRAN I compiler
- Some Algol compilers
- Lexing and parsing
- Code generation
- Intermediate languages
- Optimization
- (Flow analysis)
- (Type systems)
- (Compiler generators)
- Recommended reading and sources
- The nuclear origins of objects and classes

What is a compiler (today)?

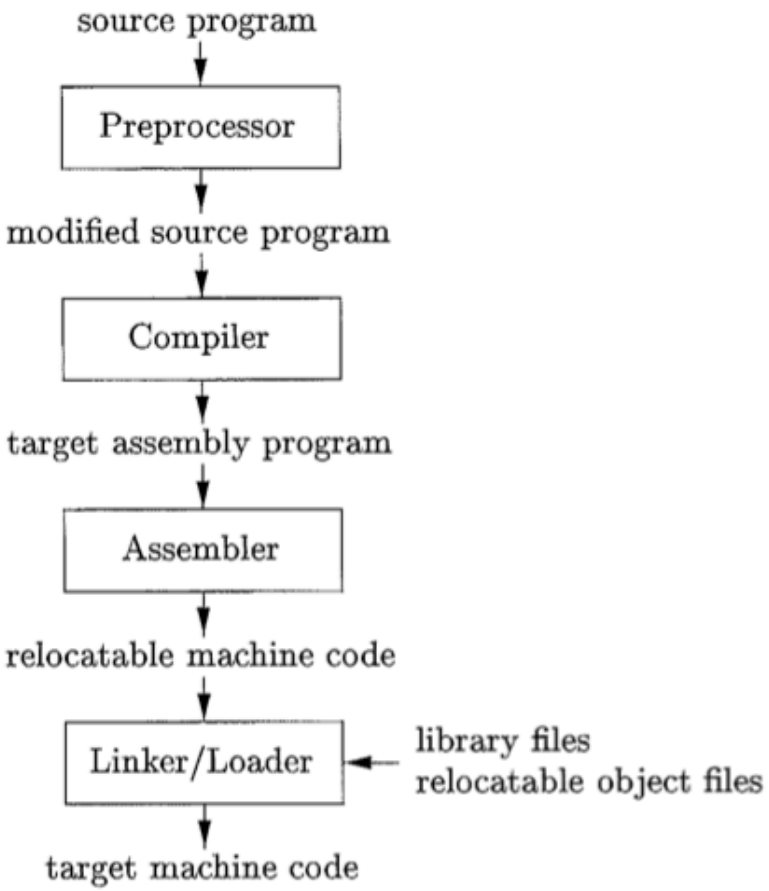
```
for (int i=0; i<n; i++)
  sum += sqrt(arr[i]);
```

C language source program

clang

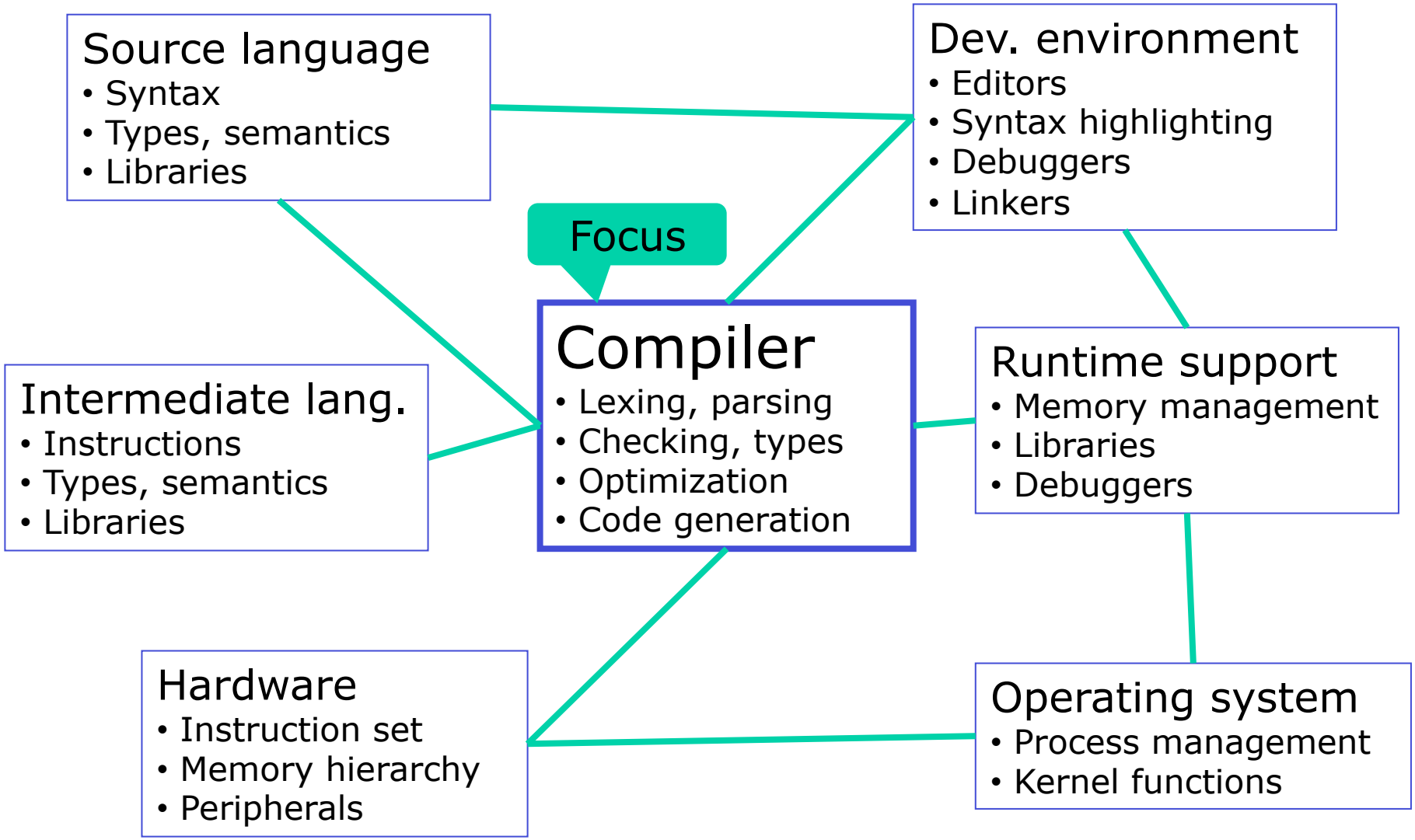
```
LBB0_1:
movl    -28(%rbp), %eax          // i
movl    -4(%rbp), %ecx           // n
cmpl    %ecx, %eax
jge     LBB0_4                   // if i >= n, return
movslq  -28(%rbp), %rax          // i
movq    -16(%rbp), %rcx          // address of arr[0]
movsd   (%rcx,%rax,8), %xmm0     // arr[i]
callq   _sqrt                    // sqrt
movsd   -24(%rbp), %xmm1         // sum
addsd   %xmm0, %xmm1             // sum + ...
movsd   %xmm1, -24(%rbp)         // sum = ...
movl    -28(%rbp), %eax          // i
addl    $1, %eax                 // i + 1
movl    %eax, -28(%rbp)          // i = ...
jmp     LBB0_1                   // loop again
```

x86 machine code

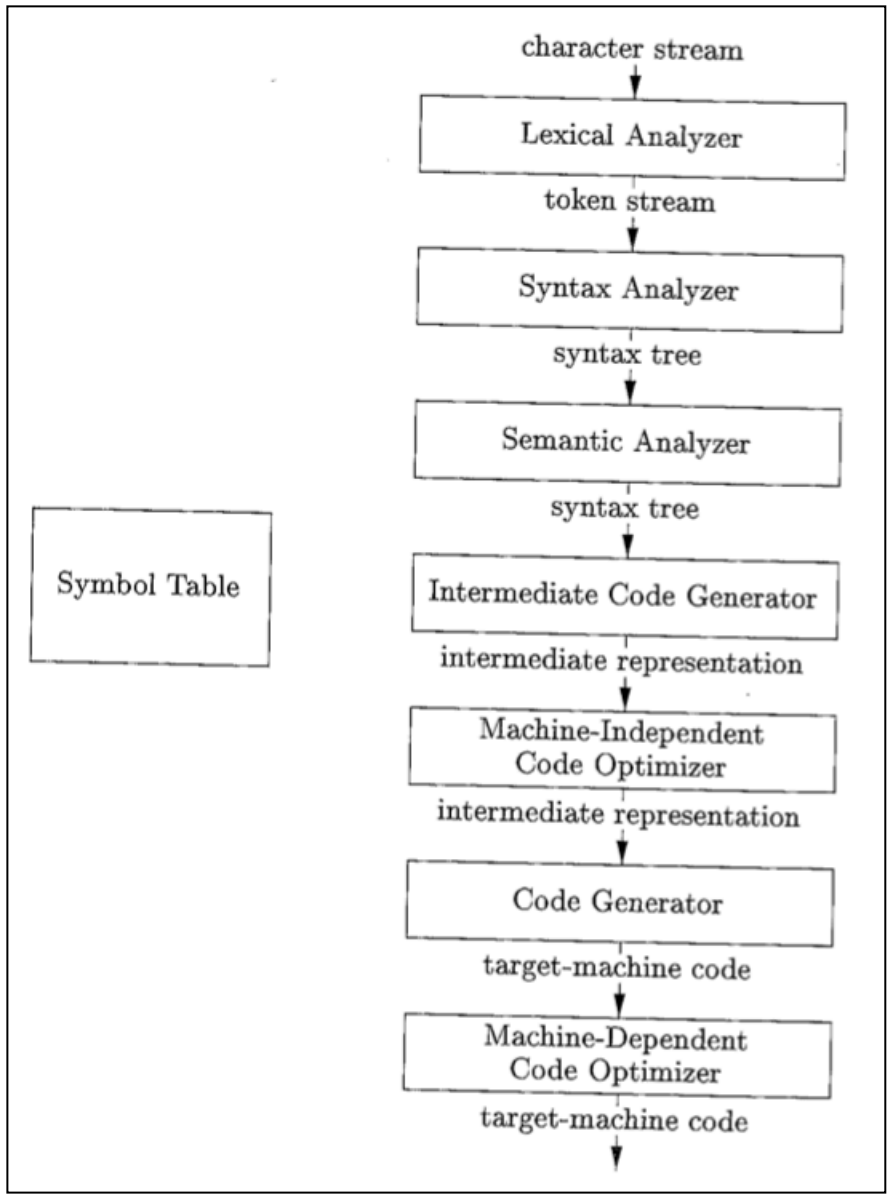


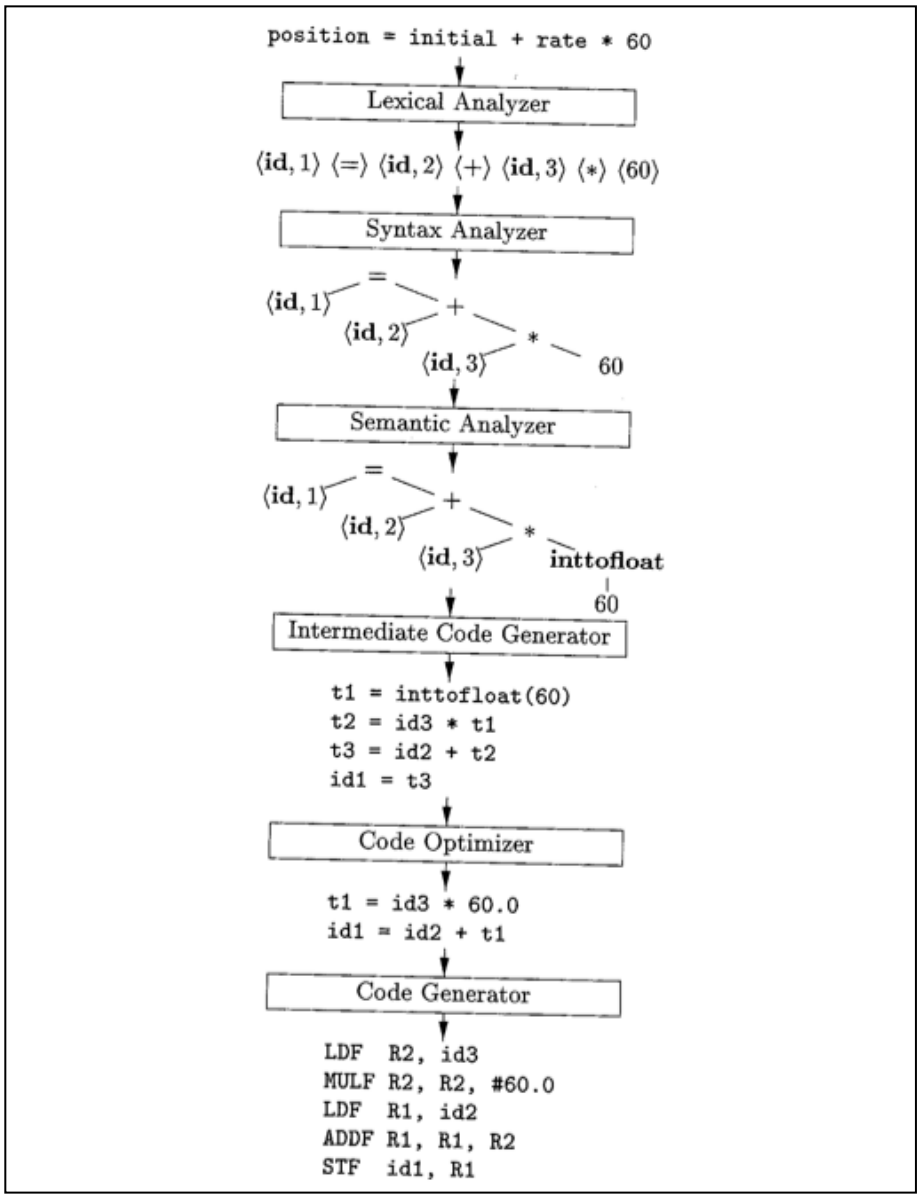
From Aho et al

A compiler in context



Conceptual phases of a compiler





From Aho et al

Why "compiler" not "translator"?

- Hopper: *A Programmer's Glossary*, 1 May 1954:

> Compile (verb) - The process of producing from pseudo-code a specific routine for a particular problem by:
>
> 1) decoding elements of information expressed in pseudo-code and segmenting the problem;
> 2) selecting or generating the required sub-routines;
> 3) transforming the subroutines into specific coding and entering them as elements in the problem routine;
> 4) maintaining a record of the subroutines used and their position in the problem routine.
>
> Compilation decodes the pseudo-code and processes static and dynamic subroutines and generators to produce a specific routine for a problem before computation.

Jones:1954:ASurvey

- Hopper's A-2 compiler collected & inlined subroutines
  - In modern terminology: *macro expansion*
- Later use: "algebraic compiler" to mean *translator*

A history of the history of ...

- Bauer: *Historical remarks on compiler construction* (1974)

  Bauer:1974:HistoricalRemarks

  - Many important references to early papers
  - USSR addendum by Ershov in 2nd printing (1976)

    Ershov:1976:Addendum

  - Opening quote:

> D. E. KNUTH [81] has observed (in 1962!) that the early history of cimpiler construction is difficult to assess. Maybe this, or maybe the general unhistorical attitude of our century is responsible for the widespread ignorance about the origins of compiler construction. In addition, the overwhelming lead of the USA in the general de-

Some older histories of ...

Knuth:1962:AHistory

- Knuth: *A history of writing compilers* (1962)
  - Few references, names and dates, mostly US:

> A complete bibliography of the compiler literature is hard to give; you may, in fact, find it quite distressing to try to read many of the articles.

> A true history gives dates of events and names of people, but I will not do that. In our field there has been an unusual amount of parallel discovery of the same technique by people working independently. Perhaps

  - Knuth later regretted this attitude

Knuth:1977:TheEarly

- Jones: *A survey of automatic coding techniques for digital computers*, MIT 1954 ! Jones:1954:ASurvey
- Randell and Russell 1964, par. 1.2 and 1.3 Randell:1964:Algol60Implementation
- Rosen: *Programming Systems and Languages*, 1964 and 1972 Rosen:1964:ProgrammingSystems Rosen:1972:ProgrammingSystems

Knuth 1977:
The early development ...

Knuth:1977:TheEarly

Language	Principal author(s)	Year	Arith-metic	Imple-men-tation	Read-abil-ity	Control struc-tures	Data struc-tures	Machine indepen-dence	Im-pact	First
Plankalkül	Zuse	1945	X, S, F	F	D	B	A	B	C	Programming language, hierarchic data
Flow diagrams	Goldstine & von Neumann	1946	X, S	F	A	D	C	B	A	Accepted programming methodology
Composition	Curry	1948	X	F	D	C	D	C	F	Code generation algorithm
Short Code	Mauchly	1950	F	C	C	F	F	B	D	High-level language implemented
Intermediate PL	Burks	1950	?	F	A	D	C	A	F	Common subexpression notation
Klammer-ausdrücke	Rutishauser	1951	F	F	B	F	C	B	B	Simple code generation, loop expansion
Formules	Böhm	1951	X	F	B	D	C	B	D	Compiler in own language
AUTOCODE	Glennie	1952	X	C	C	C	C	D	D	Useful compiler
A-2	Hopper	1953	F	C	D	F	F	C	B	Macroexpander
Whirlwind translator	Laning & Zierler	1953	F	B	A	D	C	A	B	Constants in formulas, manual for novices
AUTOCODE	Brooker	1954	X, F	A	B	D	C	A	C	Clean two-level storage
ПП-2	Kamynin & Líŭbimskiĭ	1954	F	B	C	D	C	B	D	Code optimization
ПП	Ershov	1955	F	B	B	C	C	B	C	Book about a compiler
BACAIC	Grems & Porter	1955	F	A	A	D	F	A	D	Use on two machines
Kompiler 2	Elsworth & Kuhn	1955	S	C	C	D	C	C	F	Scaling aids
PACT I	Working Committee	1955	X, S	A	C	D	C	A	C	Cooperative effort
ADES	Blum	1956	X, F	D	D	B	C	A	F	Declarative language
IT	Perlis	1956	X, F	A	B	C	C	A	A	Successful compiler
FORTRAN I	Backus	1956	X, F	A	A	C	C	A	A	I/O formats, comments, global optimization
MATH-MATIC	Katz	1956	F	B	A	C	C	A	D	Heavy use of English
Patent 3,047,228	Bauer & Samelson	1957	F	D	B	D	C	B	C	Formula-controlled computer

X=int, F=float, S=scaled

A ... F = much ... little

Other substantial secondary sources

- Naur: *The replies to the AB14 questionnaire*
  - Survey of Algol compiler construction June 1962

TABLE OF ACTIVITY REPORTS AND TRANSLATOR SPEEDS.

Brief name of group	No. of mem-bers	5: Man years	Teaching - percent -	Progr. - percent -	Impl. - percent -	12: Frac-tion per-cent	13,14: Pages written	publ.	Name of machine or system, with times for 100, 1000, and 10 000 instructions in minutes			
NPL, England	1	0.1	100	0	0	0	0	7				
Zeiss, Germany	1	3	70	20	10	-	10	-				
SMIL, Sweden	2	0.2	85	10	5	-	-	1	SMIL	0.4	4	-
Syst.Dev.Corp., USA	3	(75)	10	60	30	Note 1		0	JOVIAL	0.2	2	20

Naur:1962:Questionnaire

Naur:1962:TheReplies

- Bromberg: *Survey of programming languages and processors* (March 1963)

Bromberg:1963:SurveyOf

LANGUAGE	MANUAL				TRANSLATOR							SOURCE OF INFORMATION AND VERIFICATION	NOTES
	IDENTIFICATION	DATE	PP	PUBLISHER	CONSTRUCTOR	MACHINE	1	RUN	2	3	4	MINIMUM CONFIGURATION	
COBOL NARRATOR	95-05-000	DEC 60	161	*	RCA	RCA 501	65	SEP 60	G	A	B	16K CHAR, 6 TAPES, RDR, OFFLINE PRNTR	BROMBERG, H
COBOL NARRATOR	93-05-002	FEB 62	208	*	RCA	RCA 301	45	OCT 62	G	A	B	20K CHAR, 6 TAPES, RDR, ONLINE PRNTR	BROMBERG, H
COBOL NARRATOR		APR 62	250	ENGLISH ELECT	R.C.A.	ENGLISH ELECT. KDP10		OCT 61	G	A	B	66K, 6 TAPES, P TAPE, PRNTR	DUNCAN, FG
COBOL					SPERRY-RAND	UNIVAC II	480	OCT 60	G	A	B	2K, 12 TAPES	
COBOL 60					SPERRY-RAND	USS 80	15	APR 61	G	A	B	2 TAPES, DRUM, RDR, PCH, PRNTR	
COBOL	3166/1	OCT 60	143	*	I.C.T.	ICT 1301					B	MARK I, 2K, 12K DRUM	ELLIS, PV
					I.C.T.	ICT 1301					B	MARK II, 2K, 4 TAPES	BASED ON COBOL 60 (RAPIDWRITE VERSION IN MANUAL P155, MAY 61)
COBOL				PHILCO	COMP. SCIENCES	PHILCO 2000					B	8K, 6 TAPES	GUERNACCINI, J
COBOL	5000-21002-P	SEP 61	45	*	BURROUGHS	B-5000					B		SPEIERMAN, KH
COBOL					ARMOUR RESEARCH	UNIVAC 1103A	3.5		A	A	B	8K, 16K DRUM, 8 TAPES	BASED ON COBOL 61
COBOL 61	F-7411	JUN 61	122	NCR	NCR/GE	NCR(304)GE	50	DEC 61	A	A	AB	4.8 K, 6 TAPES, RDR, PRNTR	BASED ON COBOL 61, DUE DEC 62
													MITMAN, B
													KEATING, WP
													BASED ON COBOL 61, PLUS ELECTIVES, DUE DEC 62

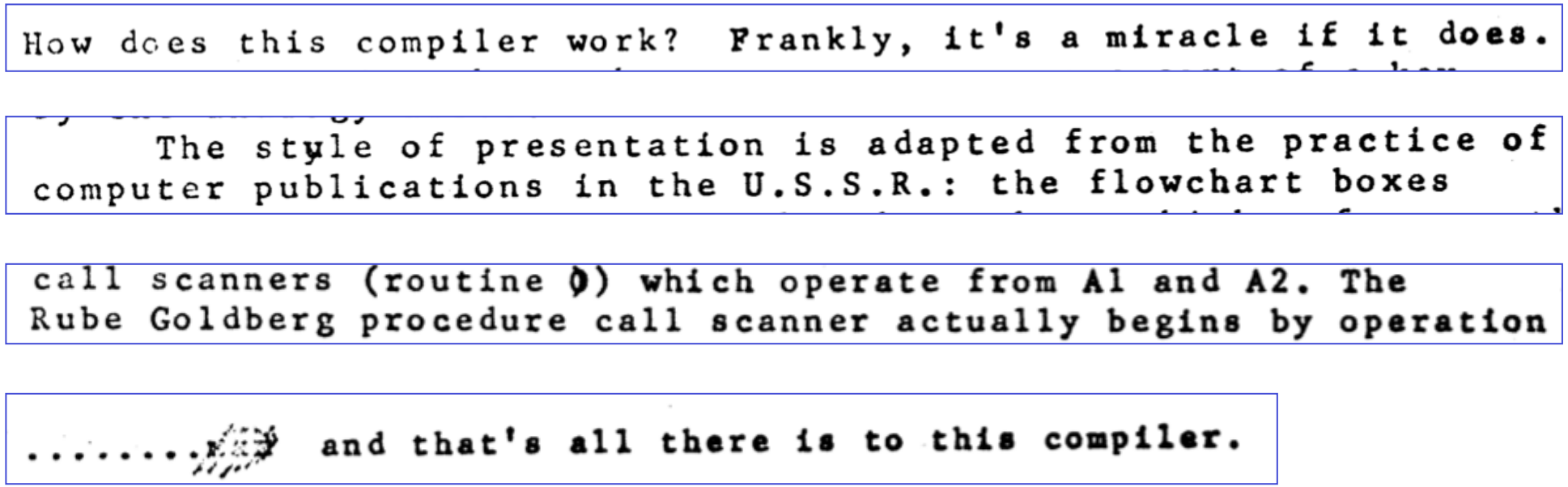
Early interpreters and compilers

Language	Machine	Operatic	Developer	Comp. size	Comp spe	Citation 1
	EDSAC	Sep-1950	Wilkes, Wheeler, Gill			Wilkes:1951:ThePreparation
Speedcode	IBM 701	Sep-1953	Backus			Backus:1954:TheIbm
A-2	Univac I	Nov-1953	Hopper			Knuth:1977:TheEarly
Autocode	Mark I	Dec-1955	Brooker			Knuth:1977:TheEarly
IT	IBM 650	Oct-1956	Perlis			Bromberg:1963:SurveyOf
Flow-Matic	Univac I	Dec-1956	Hopper			Bromberg:1963:SurveyOf
Fortran I	IBM 704	Jun-1957	Backus et al	24000 ins	8 cards/min	Backus:1957:TheFortran
Alfakod	BESK	Nov-1957	Riesel, Jonason, von Sydow			Lundin:2006:TidigProgramme
MAC	Ferranti Mercu	Dec-1957	Dahl			AMS:1958:MathematicalTable
Fortran II+III	IBM 704	May-1958	Backus et al			Backus:1959:AutomaticProgr
Runcible	IBM 650	Dec-1958	Knuth			Knuth:1959:Runcible
Algol 58	Zuse Z22	Dec-1958	Bauer, Samelson			Samelson:1960:SequentialFo
GAT	IBM 650	Feb-1959	Arden, Graham			Arden:1959:OnGat
Neliac	Univac M-460	Mar-1959	Halstead			Huskey:1959:Neliac
Algol 58	B 220	Dec-1959	Barton	3500 ins	500 instr/min	Barton:1961:AnotherNameles
Lisp 1	IBM 704	Jan-1960	McCarthy			McCarthy:1960:RecursiveFun
Algol 60	X-1	Jun-1960	Dijkstra	2500 words		Dijkstra:1960:RecursiveProgr
COBOL	RCA 501	Sep-1960	Bromberg			Bromberg:1963:SurveyOf
Algol 58	B 205	Sep-1960	Knuth	4000 words	45 cards/min	Knuth:1960:TheInternals
COBOL	Univac II	Oct-1960				Bromberg:1963:SurveyOf
Algol 60	CDC 1604	Jun-1961	Irons	800 ins/10000 tbl	300 instr/s	Irons:1961:ASyntax
Algol 60	Zuse Z22	Jul-1961	Bauer			Bromberg:1963:SurveyOf
Algol 60	DASK	Aug-1961	Jensen, Naur			Jensen:1961:AnImplementati
Algol 60	Facit EDB	Oct-1961	Dahlstrand			Dahlstrand:2009:Minnen
Algol 60	Elliott 503	Feb-1962	Hoare	8000 ins	1000 char/s	Hoare:1962:ReportOn
Algol 60	Gier	Sep-1962	Jensen, Naur	4000 words	30 instr/s	Naur:1963:TheDesign1
Algol 60 Whet	KDF9	Sep-1962	Randell, Russe	3000 words	input limited	Randell:1964:Algol60Impleme
Algol 60 Kidsg	KDF9	Dec-1962	Hawkins, Huxt	20000 words		Randell:1964:Algol60Impleme
Algol 60	M-20	Jan-1964	Ershov	45000 words		Ershov:1966:Alpha
Simula I	Univac 1107	Jan-1965	Dahl, Nygaard			Dahl:1966:Simula

Knuth's B 205 Algol 58 compiler

- Developed June-Sep 1960, at age 22
- Idiosyncratic documentation:

Knuth:1960:TheInternals
Waychoff:1979:StoriesAbout
Knuth:2007:OralHistory



How does this compiler work? Frankly, it's a miracle if it does.

The style of presentation is adapted from the practice of computer publications in the U.S.S.R.: the flowchart boxes

call scanners (routine 0) which operate from A1 and A2. The Rube Goldberg procedure call scanner actually begins by operation

..... and that's all there is to this compiler.

- No description of runtime state or its invariants

Problems and solutions

- Lexing and parsing: from text to internal representation
  - Arithmetic operators, precedence, parentheses
- Compilation of expressions
  - To postfix, reverse Polish form
- Storage allocation
  - Runtime state invariants, code to maintain them
- Optimization
  - How to generate efficient code
- Flow analysis
  - How discover program structure and invariants

History: lexing and parsing

- Initially ad hoc
- Table-driven/automata methods

Samelson:1960:SequentialFormula
Irons:1961:ASyntax
Naur:1963:TheDesign1

- Regular expressions, context-free grammars
- Finite state automata and pushdown automata
- Knuth LR parsing 1965

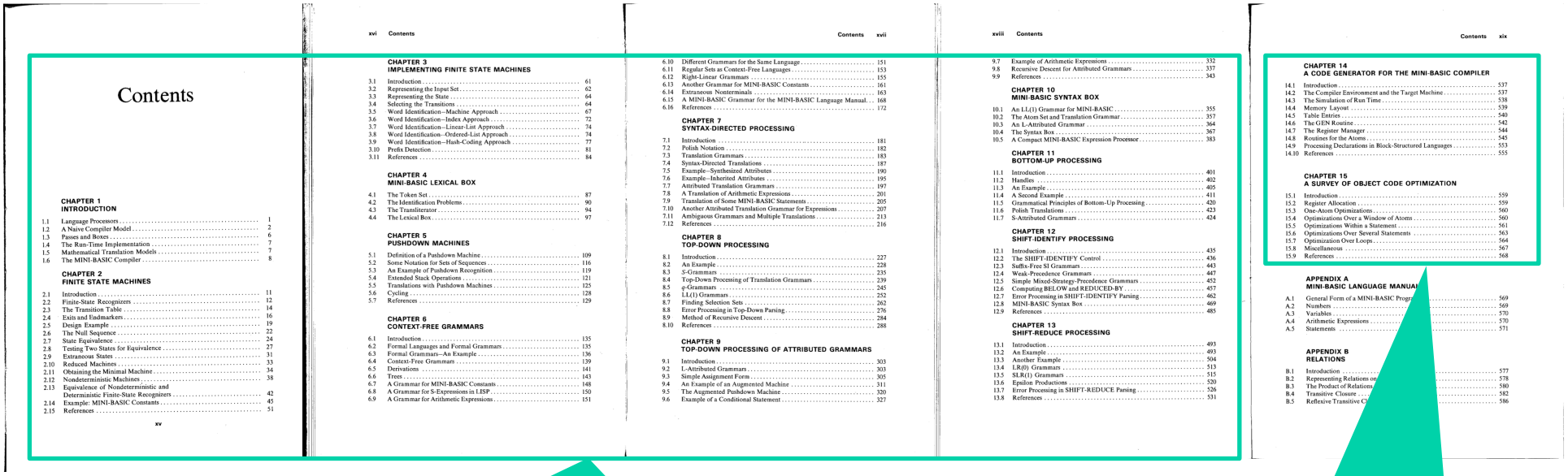
Knuth:1965:OnThe

- Gries operator grammars 1968

Gries:1968:UseOf

- Lexer and parser generator tools
  - Lex (Lesk 1975) and Yacc (Johnson 1975)
  - LR dominated for a while
  - LL back in fashion: Antlr, Coco/R, parser combinators, packrat parsers

Lewis, Rosenkrantz, Stearns: *Compiler design theory*, 1976



Contents

CHAPTER 1
INTRODUCTION

CHAPTER 2
FINITE STATE MACHINES

CHAPTER 3
IMPLEMENTING FINITE STATE MACHINES

CHAPTER 4
MINI-BASIC LEXICAL BOX

CHAPTER 5
PUSHDOWN MACHINES

CHAPTER 6
CONTEXT-FREE GRAMMARS

CHAPTER 7
SYNTAX-DIRECTED PROCESSING

CHAPTER 8
TOP-DOWN PROCESSING

CHAPTER 9
TOP-DOWN PROCESSING OF ATTRIBUTED GRAMMARS

CHAPTER 10
MINI-BASIC SYNTAX BOX

CHAPTER 11
BOTTOM-UP PROCESSING

CHAPTER 12
SHIFT-IDENTIFY PROCESSING

CHAPTER 13
SHIFT-REDUCE PROCESSING

CHAPTER 14
A CODE GENERATOR FOR THE MINI-BASIC COMPILER

CHAPTER 15
A SURVEY OF OBJECT CODE OPTIMIZATION

APPENDIX A
MINI-BASIC LANGUAGE MANUAL

APPENDIX B
RELATIONS

500 pages about lexing and parsing

30 pages **not** about lexing and parsing

- Historically, too much emphasis on parsing?
  - Because it was formalizable and respectable?
  - But also beautiful relations to complexity and computability ...

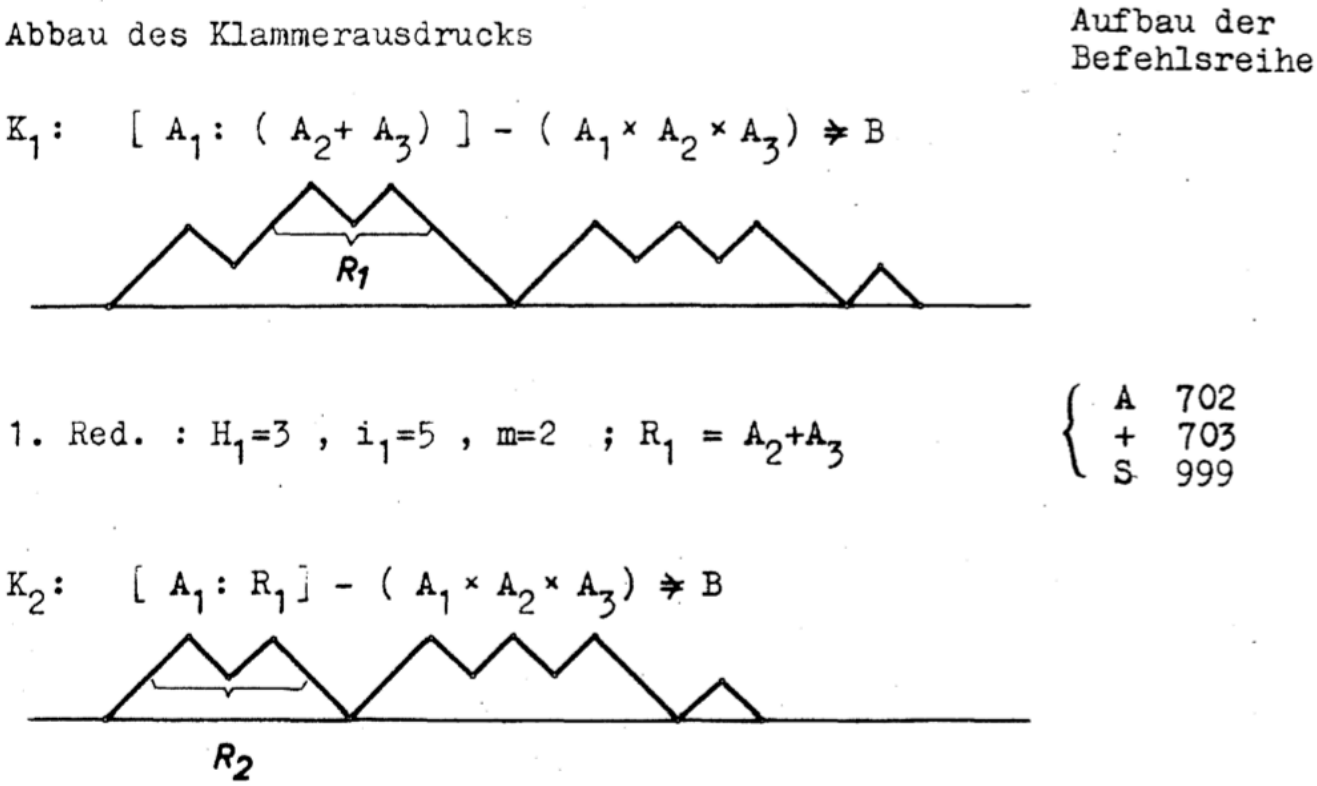
History: compilation of expressions

- Rutishauser 1952 (unimpl.) Rutishauser:1952:AutomatischeRechenplanfertigung
  - Translating arithmetic expressions to 3-addr code
  - Infix operators, precedence, parentheses
  - Repeated scanning and simplification
- Böhm 1952 (not impl.) Boehm:1954:CalculatricesDigitales Knuth:1977:TheEarly
  - Single scan expression compilation – also at ETHZ
- Fortran I, 1957 Sheridan:1959:TheArithmetic
  - Baroque but simple treatment of precedence (Böhm &)
  - Complex, multiple scans, both left-right and right-left
- Samelson and Bauer 1960 Samelson:1960:SequentialFormula
  - One scan, using a stack ("cellar") at translation time
- Floyd 1961 Floyd:1961:AnAlgorithm
  - One left scan, one right scan, optimized code

Rutishauser, ETH Zürich 1952

Rutishauser:1952:AutomatischeRechenplanfertigung

- Multi-pass gradual compilation of expression



- Seems used also by
  - Fortran-I, Sheridan 1957
  - BESM-I Programming Programme, Ershov 1958

Corrado Böhm, ETH Zürich 1951

Boehm:1954:CalculatricesDigitales

- An abstract machine, a language, a compiler
  - Three-address code with indirect addressing
  - Machine is realizable in hardware but not built
  - Only assignments $m + 1 \rightarrow m$; goto *C* is: $C \rightarrow \pi$
  - Compiler written in the compiled language
  - Single-pass compilation of full-paren. expressions

Nature r du dernier symbole

Natures de l'avant-dernier symbole	)	(	→	var.	opér.
)	G → π	Ω → π	J → π	Ω → π	T → π
(	Ω → π	C → π	Ω → π	P → π	Ω → π
→	Ω → π	Ω → π	Ω → π	Q → π	Ω → π
var.	H → π	Ω → π	Ω → π	Ω → π	W → π
opér.	Ω → π	I → π	Ω → π	R → π	Ω → π

Expression compiler transition table

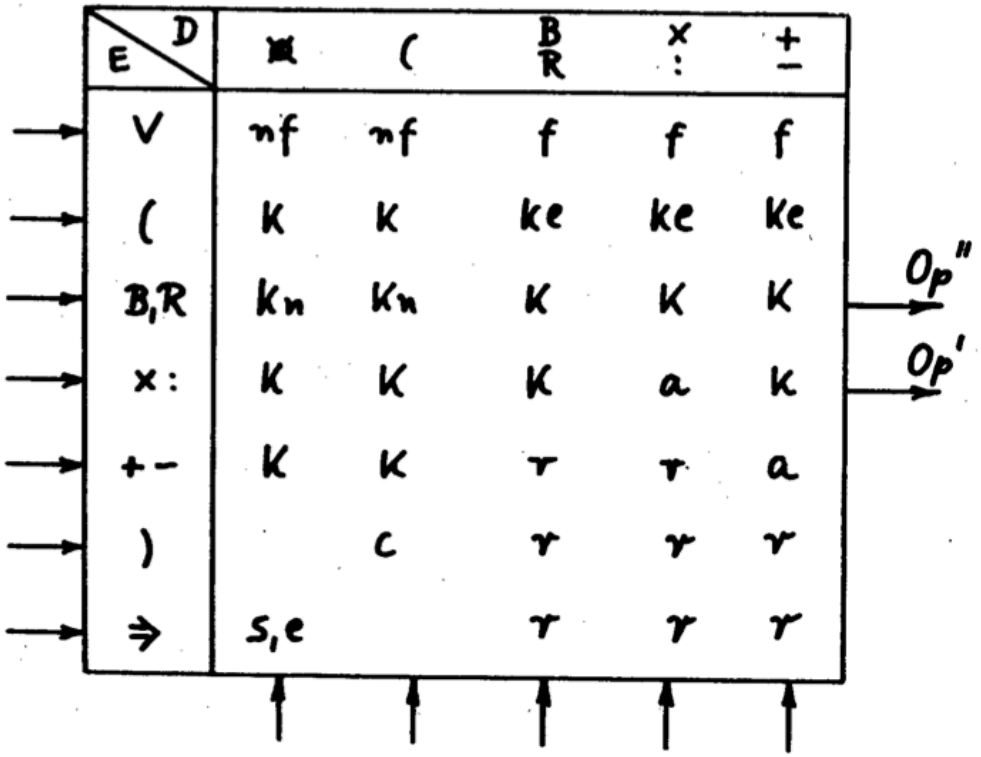
Implementation of transitions, goto

$$\pi' \rightarrow D$$
$$5 \cdot r + s + t \rightarrow \pi$$

Bauer and Samelson, Munich 1957: Sequential formula translation

- Using two stacks for single-pass translation
- Takes operator precedence into account
  - so unlike Böhm does not need full parenthetization

Bauer:1957:VerfahrenZur



Bauer and Samelson's patent

Bauer:1957:VerfahrenZur

DEUTSCHES PATENTAMT

PATENTSCHRIFT 1094 019

ANMELDETAG: 30. MÄRZ 1957

BEKANNTMACHUNG DER ANMELDUNG UND AUSGABE DER AUSLEGESCHRIFT: 1. DEZEMBER 1960

AUSGABE DER PATENTSCHRIFT: 12. AUGUST 1971

WEICHT AB VON AUSLEGESCHRIFT

1 094 019
P 10 94 019.9-53 (B 44122)

1

Die bekannten Rechenautomaten und Datenverarbeitungsanlagen erfordern im Einzelfall Anweisungen über die Art und den Ablauf der numerischen oder sonstigen informationsverarbeitenden Prozesse. Die Schreibweise, in der diese Anweisungen fixiert werden, wurde zu Beginn der Entwicklung so gewählt, daß sie gewisse als elementar erachtete technische Funktionen der Anlage beschrieb. Die so geschriebenen Anweisungen werden üblicherweise »Programm« genannt. Das Programm für einen Rechenprozeß etwa und die mathematische Formel, mit der der Mathematiker diesen Prozeß gewöhnlich be-

Rechenmaschine zur automatischen Verarbeitung von kodierten Daten

Patentiert für:

Siemens AG,
1000 Berlin und 8000 München

Dr. Friedrich Ludwig Bauer
und Dr. Klaus Samelson, 8000 München,
sind als Erfinder genannt worden

History: Compilation techniques

- Single-pass table-driven with stacks
  - Bauer and Samelson for Alcor
  - Dijkstra 1960, Algol for X-1 Dijkstra:1961:Algol60Translation
  - Randell 1962, Whetstone Algol Randell:1964:WhetstoneAlgol
- Single-pass recursive descent Hoare:1962:ReportOn
  - Hoare 1962, one procedure per language construct
- Multi-pass ad hoc
  - Fortran I, 6 passes Backus:1957:TheFortran
- Multi-pass table-driven with stacks
  - Naur 1962 GIER Algol, 9 passes Naur:1963:TheDesign2
  - Hawkins 1962 Kidsgrove Algol
- General syntax-directed table-driven
  - Irons 1961 Algol for CDC 1604 Irons:1961:ASyntax

Multi-pass compilation is still used

- Good for small-memory systems (eg GIER)
- Still used, now for separation of concerns:

21 internal passes of the Scala compiler (2013)

namerFactory
typerFactory
superAccessors
pickler
refchecks
liftcode
uncurry
tailCalls
explicitOuter
erasure
lambdaLift
constructors
flatten
mixer
cleanup
genicode
inliner
inlineExceptionHandlers
closureElimination
deadCode
genJVM

History: Run-time organization

- Early papers focus on *translation*
  - Runtime data managent is trivial, eg. Fortran I
- Algol: *runtime storage allocation* is essential
- Dijkstra: Algol for X-1 (1960) Dijkstra:1960:RecursiveProgramming
  - Stack of procedure activation records
  - Display, to access variables of enclosing scopes
- Also focus of Naur's Gier Algol papers, and Ekman's thesis on SMIL Algol Naur:1963:TheDesign1 Naur:1963:TheDesign2 Ekman:1962:KonstructionOch

- Design a runtime state structure (invariant)
- Compiler should generate code that
  - Can rely on the runtime state invariant
  - Must preserve the runtime state invariant

Diskussion om ALGOL vid symposiet i Rom i mars 1962. Sittande fr.v. P. Naur, Danmark, S. Moriguti, Japan, och A. van Wijngaarden, Nederländerna. Stående E. W. Dijkstra, Nederländerna.

Ekman and Fröberg 1972, p. 127

History: Target architectures

- EDSAC, IAS machine, BESK
  - No index register, so self-modifying code for arrays
  - Subtle, and makes manual code relocation painful
- Index registers: Manchester Mark 1, DASK, ...
  - Simpler and faster code, position-independent
  - IBM 704 at-least-once loops impacts Fortran DO
- Stack machines
  - Conceptual: Samelson, Hamblin, Dijkstra, Barton
  - Hardware: Burroughs B5000, KDF9 (arith, return)
  - Compile to reverse Polish by post-order traversal
  - Java and .NET/CLI intermediate languages

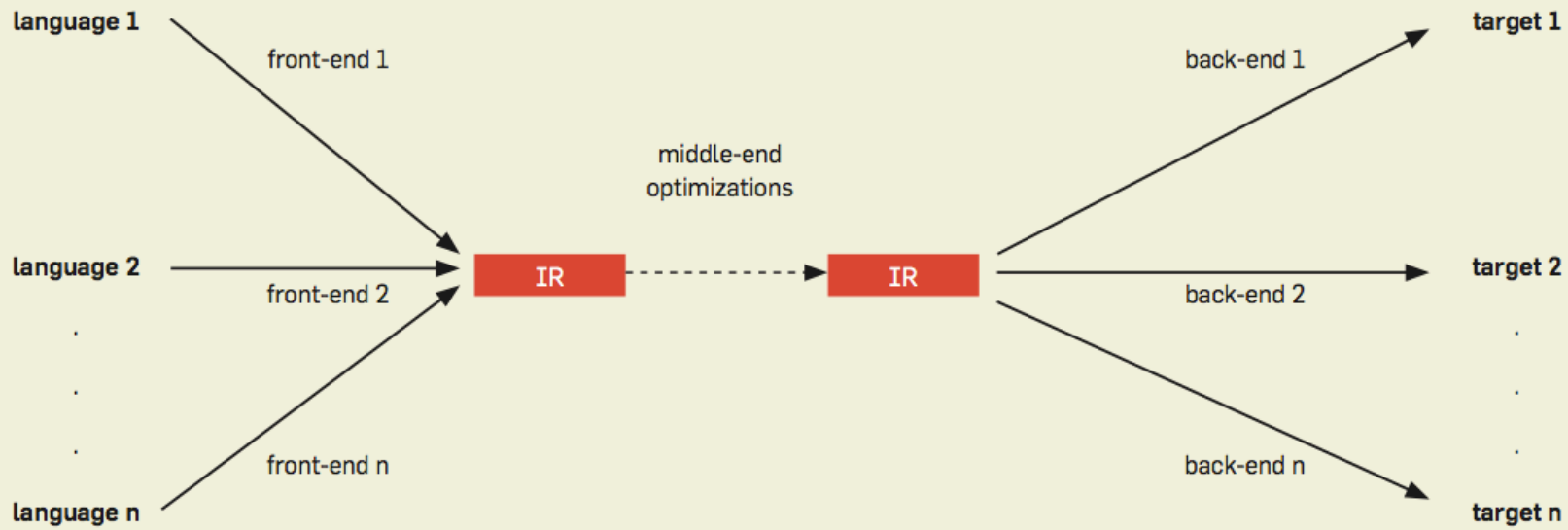
Hamblin:1962:TranslationTo
Dijkstra:1960:RecursiveProgramming
Barton:1961:ANew

Everything old is new again

- Chow: *Intermediate representation*, CACM December 2013

Chow:2013:IntermediateRepresentation

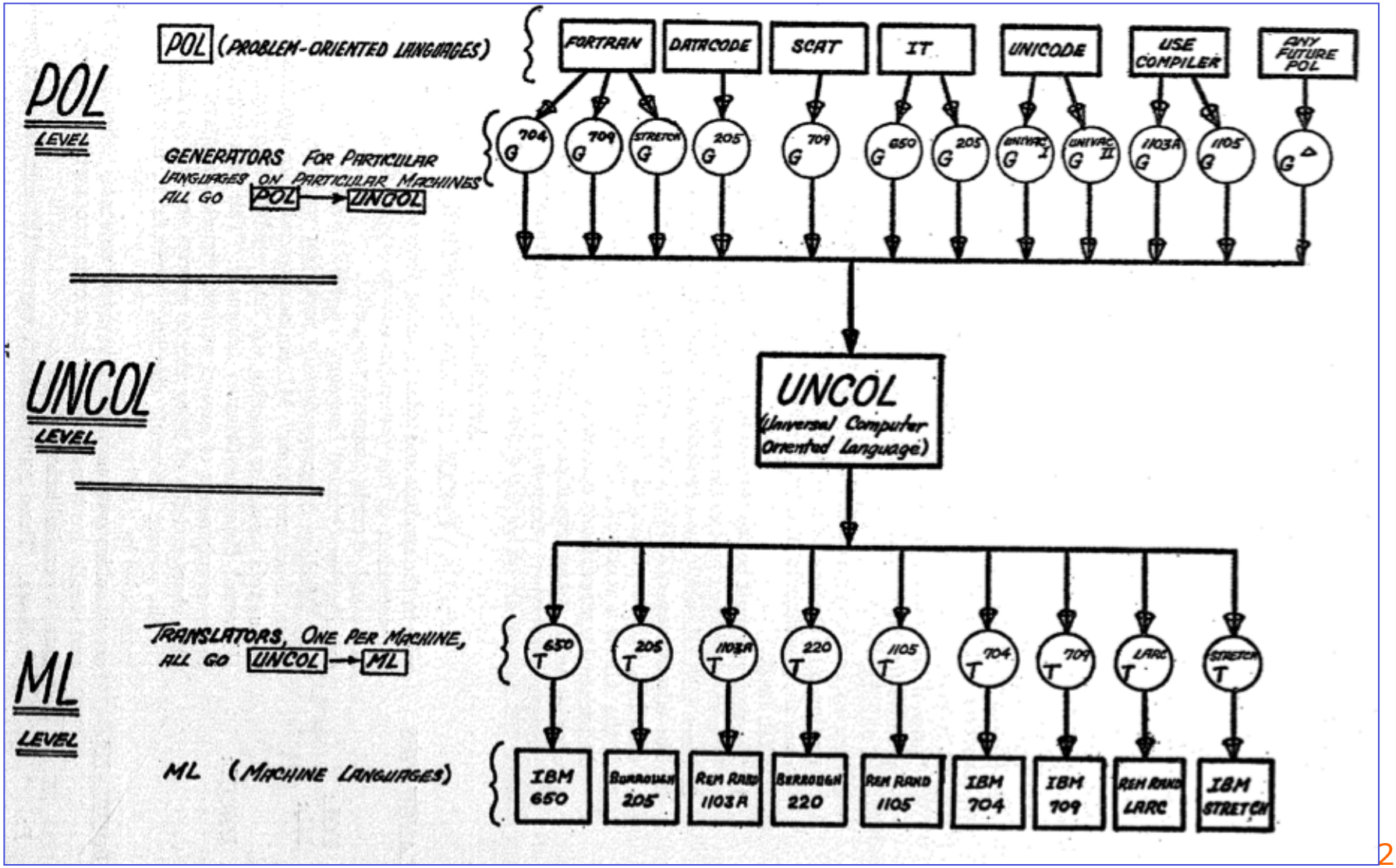
Figure 2. A compiler system supporting multiple languages and multiple targets.



History: Intermediate languages

Strong:1958:TheProblem

- Strong: *The problem of ...*, February 1958



UNCOL is finally coming true

- Java Virtual Machine, 1994
  - Developed as bytecode for Java only
  - Yet used for Scala, Clojure, Jython, Ceylon, JRuby...
  - Runtime system, libraries, on many CPUs and OSs
- .NET Common Language Infrastructure, 1999
  - C#, VB.NET, F#, JScript, Eiffel, COBOL, IronPython
- JavaScript/EcmaScript, 1994
  - For browser scripting, thus on all user devices
  - Ceylon, Dart, Typescript, F#, ...
- LLVM = Low-Level Virtual Machine, 2003
  - Static single assignment compiler-internal form
  - Clang C/C++/Objective-C, Nvidia CUDA, Mono, ...

What does a C# compiler do

```
for (int i=0; i<n; i++)
  sum += sqrt(arr[i]);
```

C# language source program

csc /o

```
0b stloc.1              // i = 0
0c br.s 1d              // goto 1d
0e ldloc.0              // sum
0f ldarg.1              // arr
10 ldloc.1              // i
11 ldelem.r8            // arr[i]
12 call Math::Sqrt      // sqrt
17 add                  // sum + ...
18 stloc.0              // sum
19 ldloc.1              // i
1a ldc.i4.1             // 1
1b add                  // i + 1
1c stloc.1              // i = ...
1d ldloc.1              // i
1e ldarg.0              // n
1f blt.s 0e             // if i<n loop
```

.NET/CLI bytecode

runtime system's
just-in-time compiler

```
19 xorl %ebx,%ebx                  // i = 0
1b jmp 3a                          // goto 3a
1d leal 0x00(%ebp),%ebp
20 fldl 0xec(%ebp)                 // sum
23 cmpl %ebx,0x0c(%edi)            // array index check
26 jbe 49                          // if outofbounds, throw
2c leal 0x10(%edi,%ebx,8),%eax     //
30 fldl (%eax)                     // arr[i]
32 fsqrt                           // sqrt
34 faddp %st,%st(1)                // sum + ...
36 fstpl 0xec(%ebp)                // sum = ...
39 incl %ebx                       // i++
3a cmpl %esi,%ebx
3c jl 20                           // if i<n, loop
```

x86 machine code

.NET/CLI bytecode:

- Stack-based
- Loadtime checks, types, local stack
- Runtime checks, array bounds, null references, ...
- Dynamic compilation to real machine code

History: Type checking

- Naur GIER Algol 1965 Naur:1965:CheckingOf
  - "*pseudoevaluation of the expressions*"
  - "*like a run-time evaluation but works with descriptions of the types and kinds of operands instead of with values*"
- Damas and Milner, ML 1982 Damas:1982:PrincipalTypeschemes
  - Inference of polymorphic type schemes
  - Generalization and specialization
  - Unification (Robinson) to solve type equations
  - Has influenced Haskell, C#, F#, Scala, ...

Compiler quality attributes

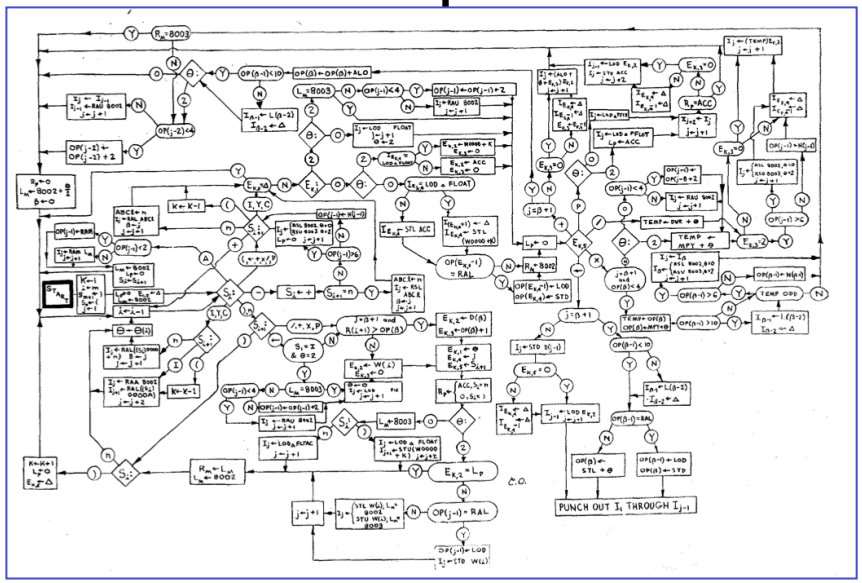
- Produces fast target code
- Produces target code fast
- Checks source code syntax
- Checks source code types and consistency
- Provides precise and clear error messages
- Reports as many errors as possible
- Is itself free of errors
- Does not report spurious errors
- Provides frugal compilation or recompilation or separate compilation

History: Diagnostics

- EDSAC (Wilkes 1951)
  - Tracing and post-mortem dumps
- GIER Algol
  - Comprehensive compiler error messages
  - Not a focus in eg Hoare's Elliot Algol compiler
  - "*GIER ... had an excellent compiler-cum-operating system*" (Sanders in HiNC 2003)
  - "*the very successful Algol compiler ... for the GIER computer*" (Randell & Russell 1964)
  - (Presumably also important that it was thoroughly tested, Naur:1963:TheDesign2 page 163)

History: Bootstrapping

- Writing a compiler in the language it compiles
- Runcible expression compilation chart, in Runcibl



Knuth:1959:Runcible

Huskey:1959:Neliac

Masterson:1960:CompilationFor

- Neliac compiler written in Neliac 1959
  - and self-compilation as correctness check
- EASY-FOX assembler 1955

Gruenberger:1968:TheHistory

- Böhm's theoretical compiler 1951

Boehm:1954:CalculatricesDigitales

C compiler optimizations

```
for (int i=0; i<n; i++)
  sum += sqrt(arr[i]);
```

C language

clang

```
LBB0_1:
movl    -28(%rbp), %eax          // i
movl    -4(%rbp), %ecx           // n
cmpl    %ecx, %eax
jge     LBB0_4                   // if i >= n, return
movslq  -28(%rbp), %rax          // i
movq    -16(%rbp), %rcx          // address of arr[0]
movsd   (%rcx,%rax,8), %xmm0     // arr[i]
callq   _sqrt                    // sqrt
movsd   -24(%rbp), %xmm1         // sum
addsd   %xmm0, %xmm1             // sum + ...
movsd   %xmm1, -24(%rbp)         // sum = ...
movl    -28(%rbp), %eax          // i
addl    $1, %eax                 // i + 1
movl    %eax, -28(%rbp)          // i = ...
jmp     LBB0_1                   // loop again
```

clang -O3

x86 machine code

```
LBB0_2:
movsd   (%rsi), %xmm1            // *p same as arr[n-i]
sqrtsd  %xmm1, %xmm1             // sqrt
addsd   %xmm1, %xmm0             // sum += ...
addq    $8, %rsi                 // p++
decq    %rax                     // i--
jne     LBB0_2                   // if i!=0 loop again
```

- Register allocation (sum, i, n, arr)
- Inlining of functions
- Index calculations
- Iteration variable elimination (i)

History: Optimization

- Fortran I in 1957 has Backus:1957:TheFortran
  - common subexpression elimination
  - fast index computations: reduction in strength
  - clever allocation of index registers
  - constant folding
- Samelson & Bauer Samelson:1960:SequentialFormula
  - algorithm for fast index computations (red.stren.)
- Allen 1969, algorithms for: Allen:1969:ProgramOptimization
  - basic blocks, flow graph of strongly conn. comps.
  - constant folding
  - common subexpression elimination
  - invariant code moving
  - reduction in strength
  - test replacement (for loops after red. strength)

Ershov 1957: Optimizations in PP BESM

Ershov:1980:TheEarly

- Fast common subexpression elimination
  - Based on hashing, independently invented
- Reuse of memory locations for intermediate results, by backwards analysis
- With Lavrov (1962): Graph coloring for register allocation

History: Flow analysis

- Fortran I was amazingly ambitious Backus:1957:TheFortran
  - Control flow analysis graph with edge frequencies
  - Monte Carlo simulation at compiletime based on FREQUENCY statements
- Allen, Cocke 1970 Allen:1970:ControlFlow
  - Control flow graph, dominator, interval, reducible graph, ...
  - Left out of Bauer:1974:HistoricalRemarks
- Cousot and Cousot 1977
  - Abstract interpretation
  - Based on formal semantics
  - Systematic approximation via Galois connections
  - Termination via widening operators

Compiler compilers or generators

- 1961:

OrchardHays:1961:TheEvolution

> A notion that has been considered for the past several years is that of a compiler for writing compilers.

- Feldman, Gries survey 1968

Feldman:1968:TranslatorWriting

  - Both syntax (parsing) and semantics

- Semantics-Directed Compiler Generation
  - January 1980 in Aarhus; LNCS 94, ed. Neil Jones
  - From formal semantics to compiler
    - or from interpreter to compiler ≈ partial evaluation

Some topics not covered at all

- Adaptive compilation
  - just-in-time compilers for Java, .NET/CLI
- Compilation techniques for
  - lazy functional languages: Haskell
  - object-oriented languages: Java, C#
  - dynamic languages: Smalltalk, Javascript (v8)
  - extensible languages: Lua
- Compilation for parallel architectures
- Continuation-based compilation
- ...

Interesting reading

- Secondary sources
  - Knuth:1977:TheEarly
  - Bauer:1974:HistoricalRemarks
  - Ershov:1976:Addendum
  - Randell:1964:Algol60Implementation sec 1.2, 1.3
- Primary sources
  - Backus:1957:TheFortran
  - Samelson:1960:SequentialFormula
  - Dijkstra:1960:RecursiveProgramming
  - Hoare:1962:ReportOn
  - Naur:1963:TheDesign1
  - Naur:1965:CheckingOf
  - Randell:1964:Algol60Implementation

A collection of sources (excerpt)

3								
4	**Bibtex and link**	**Author**	**R**	**C**	**Title**	**Notes, message**	**Category**	**Source**
5	Aiken:1946:TheAutomatic	Aiken, Hopper		US	The automatic sequence contro	The IBM Harvard Mark-I, electromechanical, not a store	hardware	Electrical Engine
6	AlgolBulletin			W	Algol Bulletin		compilers	At http://archive
7	Allen:1969:ProgramOptimi	Allen		US	Program optimization		static analysis	Annual review in
8	Allen:1970:ControlFlow	Allen		US	Control flow analysis		static analysis	ACM Sigplan Not
9	Amdahl:1989:OralHistory	Amdahl		US	An interview with Gene Amdah	Brief mention of WISC computer (1951) on page 28ff,	history	At http://conser
10	AMS:1958:MathematicalTa	American Mathematica		US	Review of Dahl:1957:AutocodingFor		compilers	Mathematical Ta
11	Andersen:1958:LaereboqI	Andersen, Bech, Mølle		DK	Lærebog i kodning for DASK	40-bit words, 20 bit instructions = 7 op bits + 11 addr	hardware	At http://datam
12	Arden:1959:OnGat	Arden, Graham		US	On GAT [Generalized Algebraic	Translator] and the construction of translators	compilers	CACM 2, 7 (July
13	Asker:2005:AlgolGenius	Asker		SE	Algol-Genius: An early success	for high-level languages	languages	Bubenko:2005:H
14	Backus:1954:TheIbm	Backus		US	The IBM 701 Speedcoding syst	Apparently no compilation process, input is three-adde	interpreters	JACM 1, 1 (1954
15	Backus:1957:TheFortran	Backus et al		US	The FORTRAN automatic codin	About Fortran I, early 1957; only simple arithmetic fun	languages	Western comput
16	Backus:1959:AutomaticPro	Backus		US	Automatic programming: Prope	About Fortran II; adds general procedures and functior	compilers	In NPL:1959:Me
17	Barton:1961:ANew	Barton		US	A new approach to the functior	Algol 60 inspired machine design with stack S and regi	hardware	AFIPS Conferenc
18	Barton:1961:AnotherName	Barton		US	Another (nameless) compiler fc	Some statistics on the Burroughs 220 Algol-58/60 com	compilers	CACM 4, 1 (Janu
19	Bauer:1957:VerfahrenZur	Bauer, Samelson		DE	Verfahren zur automatischen V	Machine (hardware) patent on one-pass formula compi	compilers	depatisnet.dpma
20	Bauer:1959:SequentielleFo	Bauer, Samelson		DE	Sequentielle formelübersetzung		compilers	Elektronische Re
21	Bauer:1974:HistoricalRema	Bauer	rere	DE	Historical remarks on compiler	With bibliography (which misses Allen, Cocke)	compilers	Bauer et al: Com
22	Bemer:1969:PoliticoSocial	Bemer		US	A politico-social history of Algol		languages	Annual Review o
23	Bergin:1996:HistoryOf	Bergin, Gibson		W	History of Programming Languages II		history	ACM Press 1996
24	Bergin:2007:AHistory	Bergin		W	A history of the history of programming languages		languages	CACM 50, 5 (Ma
25	Bitsavers::PdfArchive	Bitsavers		W	PDF document archive	Roughly 28000 scanned documents, books, manuals ar	history	At http://bitsave
26	Boehm:1954:CalculatricesI	Böhm		CH	Calculatrices digitales: du déch	Describes theoretical three address machine enabling i	compilers	Diss. Math. ETH
27	Bratman:1961:AnAlternate	Bratman		US	An alternate form of the UNCO	First appearance of T diagrams	compilers	CACM 4, 3 (Marc
28	Bromberg:1963:SurveyOf	Bromberg		W	Survey of programming languages and processors		compilers	CACM 6, 3 (Marc
29	Bruderer:2013:WarumWoll	Bruderer		CH	Warum wollte Konrad Zuse 194	Speculates that Zuse might be afraid of abduction to U	history	At http://e-colle
30	Bubenko:2005:HistoryOf	Bubenko, Impagliazzo,		SC	History of Nordic computing 2003		systems	Springer 2005
31	Burks:1946:PreliminaryDis	Burks, Goldstine, von		US	Preliminary discussion of the lo	Describes what became the Princeton IAS computer (a	hardware	At http://library.
32	Carr:1959:AVisit	Carr et al		SU	A visit to the computation cent	Mentions an automatic programming system in Moscov	tripreport	CACM 2, 6 (1959
33	Chow:2013:IntermediateR	Chow		W	Intermediate representation		compilers	CACM 56, 12 (D
34	ComputingLab:1960:WiscU	Computing Lab		US	Wisc users' manual		hardware	At http://bitsave
35	Conway:1958:ProposalFor	Conway		US	Proposal for an UNCOL	A concrete proposal for the UNCOL concept suggested	intermediate	CACM 1, 10 (195
36	Dahl::APlea	Dahl		NO	A plea for multiprogramming	Identifies three kinds of parallelism (multiple processor	concurrency	Algol Bulletin 24
37	Dahl:1957:AutocodingFor	Dahl		NO	Autocoding for the Ferranti Mer	Declarations of types (eg multidimensional array layout	compilers	NDRE Report 24
38	Dahl:1957:Errata	Dahl		NO	Errata page for: Autocoding for	Errata for Dahl:1957:AutocodingFor; indicates some m	compilers	From Knut, Oslo
39	Dahl:1957:MultipleIndex	Dahl		NO	Multiple index countings on the	Storage and efficient indexing into multidimensional ar	compilers	NDRE Report 23
40	Dahl:1958:ProgrammersHa	Dahl, Garwick		NO	Programmer's handbook for the	The machine has 40 bit long words = 2 x 20 bit registe	hardware	Second edition 1
41	Dahl:1966:Simula	Dahl, Nygaard		NO	Simula, an Algol-based simulation language		compilers	CACM 9, 9 (Sept
42	Dahl:1970:CommonBase	Dahl, Myhrhaug, Nyga		NO	Common Base Language		languages	Simula Informat
43	Dahl:2002:TheBirth	Dahl		NO	The birth of object orientation:	the Simula languages	languages	At http://www.o
44	Dahlquist:1956:KodningFo	Dahlquist et al		SE	Kodning för BESK	40-bit words, 20 bit instructions = 7 op bits + 11 addr	hardware	At http://user.it.
45	Dahlstrand:2009:Minnen	Dahlstrand		SE	Ingemar Dahlstrands Minnen	Wrote Algol 60 compiler for BESK and Facit; see pages	compilers	At http://www.t
46	Damas:1982:PrincipalType	Damas, Milner		UK	Principal type-schemes for func	Polymorphic type inference	types	POPL 1982, 207
47	Davis:1978:TheSoviet	Davis, Goodman		SU	The Soviet bloc's unified system of computers		hardware	Computing Surv

The nuclear origins of OO

- Garwick, Nygaard, Dahl at NDRE = Norwegian Deference Research Establishment
- Norway 6th country to have a nuclear reactor
  - in November 1951
  - six years before Risø, and 30 times as powerful
- Garwick and Nygaard computed parts of the reactor design 1947-1951
  - w Monte Carlo methods to simulate neutron flow
  - chiefly hand calculators
- O.-J. Dahl hired 1952
  - developed "programs" for modified Bull mech. calc.
  - from 1957 developed MAC autocoding for Ferranti

Norway: Nuclear and computing 1946-1962

Holmevik:2005:InsideInnovation

Forlan:1997:NorwaysNuclear

Forlan:1987:PaaLeiting

Randers:1946:RapportTil

Norway's nuclear and computing pioneers 2014-02-06

	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	...
Inst. Astrophysic	GRan																									
	JGar	JGar	JGar	JGar			JGar	JGar																		
		KNyg																								
NDRE/IFA								GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	GRan	
								ODah	ODah	ODah	ODah	ODah	ODah	ODah	ODah											
									JGar	JGar	JGar	JGar	JGar	JGar	JGar	JGar	JGar	JGar					JGar	JGar	JGar	
										KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg	KNyg					
														OJDa	OJDa	OJDa	OJDa	OJDa	OJDa	OJDa	OJDa	OJDa	OJDa			
NCC																						KNyg	KNyg	KNyg	KNyg	
																								OJDa	OJDa	
USA		GRan	GRan	GRan				GRan																		
								ODah																		
UK					GRan	GRan	GRan																			
France									JGar																	
							rep		nuc	nuc	nuc	nuc	crit						mac	mac	mac			sim 1	sim 1	
Gunnar Randers	GRan																									
Odd Dahl	ODah																									
Jan Garwick	JGar																									
Kristen Nygaard	KNyg	b																								
Ole-Johan Dahl	OJDa	g																								

Thanks to

- Christian Gram, Datahistorisk Forening
- Robert Glück, DIKU
- Birger Møller-Petersen, Oslo University
- Knut Hegna, Oslo University Library
- Bjørg Asphaug, NDRE Library
- Ingemar Dahlstrand, Lund University
- Torgil Ekman, Lund University
- Mikhail Bulyonkov, Russ. Ac. Sci. Novosibirsk
- Christine di Bella, IAS Archives, Princeton
- George Dyson, Bellingham WA, USA