Introduction to Program Synthesis (WS 2024/25)

Chapter 1 - Introduction

Dr. rer. nat. Roman Kalkreuth

Chair for AI Methodology (**AIM**), Department of Computer Science,
RWTH Aachen University, Germany

An early approach to PS

Instruction tables will have to be made up by mathematicians with computing experience and perhaps a certain puzzle-solving ability. This process of **constructing instruction tables should be very fascinating***. There need be no real danger of it ever becoming a drudge, for any processes that are quite mechanical* **may be turned over to the machine itself***.* - **Alan Turing** (in *Proposed Electronic Calculator* (1945))

An early approach to PS
The Fortran I compiler

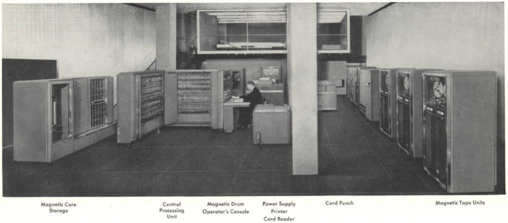
- In 1954 one of the first compiler for Fortran (*Formula Translation*) was introduced
  - First commercial release in 1957
- Fortran is one of the oldest programming languages
  - Specifically designed for scientific numerical computing
- Translation from Fortran programs to assembly code for the IBM 704 mainframe

The FORTRAN Automatic Coding System

J. W. BACKUS†, R. J. BEEBER†, S. BEST‡, R. GOLDBERG†, L. M. HAIBT†, H. L. HERRICK†, R. A. NELSON†, D. SAYRE†, P. B. SHERIDAN†, H. STERN†, I. ZILLER†, R. A. HUGHES§, AND R. NUTT‖

An early approach to PS
The Fortran I compiler

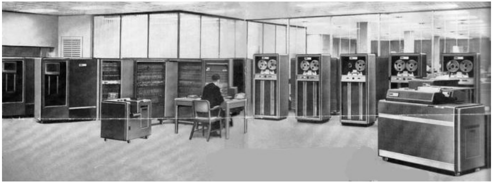
- IBM 704 mainframe → first mass-produced computer capable of handling floating-point arithmetic
  - 40,000 instructions per second → up to 12,000 floating-point additions per second
  - Instruction set size: 102 instructions
- Magnetic Core Storage Unit → 4,096 × 36-bit words (18,432 bytes) of RAM
- FORTRAN and LISP were the first to be developed for the IBM 704

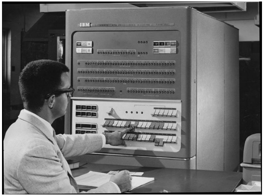


IBM 704 ELECTRONIC DATA-PROCESSING MACHINES

An early approach to PS
The Fortran I compiler

- Data registers: **Accumulator** (38 bit), **multiplier/quotient register** (36-bit)
- Three 15-bit **index registers**
- 15-bit **program counter**





An early approach to PS
The Fortran I compiler

- Demonstration that it is possible to automatically generate efficient machine code from a high-level language
  - First known successful attempt in code optimization
- Considered as one of the 10 algorithms with the greatest influence in science and engineering in the 20th century
  - Computing in Science & Engineering Journal (2000) [DS00]

- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

Nostalgic Programming: Fortran
Historical Sketch (Fortran Standards)

- 1957 • John Backus at IBM is developing a language for *Formula translation* → **Fortran I**
- 1966 • First ANSI[1] standard for a programming language ever →**Fortran 66**
- 1978 • ANSI standard X3.9-1978 defines **Fortran 77**
- 1992 • ANSI standard X3.198-1992 defines **Fortran 90**
- 1997 • ISO/IEC 1539-1:1997 revises (and extends) Fortran 90 to **Fortran 95**
- 2004 • ISO/IEC 1539-1:2004 introduces **Fortran 2003**
- 2010 • **Fortran 2008** (ISO/IEC 1539-1:2010)
- 2018 • **Fortran 2018** (ISO/IEC TS 29113:2012 & ISO/IEC TS 18508:2015)
- 2023 • **Fortran 2023** (ISO/IEC 1539-1:2023)

[1] American National Standards Institute (https://www.ansi.org/)

Nostalgic Programming: Fortran
Historical Sketch (Fortran Features)

1957	I/O, DO loops, GOTO, IF statements
1958	Functions
1966	Booleans, portability
1977	Block if/else, strings, I/O revision
1990	Free-form input, parallelism (SIMD), recursion, memory allocation, modules
1995	SIMD parallelism revision
2003	Object-oriented programming, function pointers
2008	Further revision and extension (SIMD and MIMD)
2018	Additional features for parallelism
2023	Correcting errors and omissions from Fortran 18

Nostalgic Programming: Fortran

Fun Facts

- Fortran is older than Professor Hoos
- Fortran ranks #9 in the π approximation speed comparison benchmark[2]
- Fortran II compiler for the PDP-8 $\rightarrow$ 3700 12-bit words of memory
  - PDP-8 had 4096 $\times$ 12-bit words of memory (6 Kilobytes)
  - GCC (GNU Compiler Collection) had over 14 million lines of code in 2015

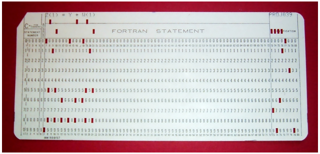
[2] https://github.com/niklas-heer/speed-comparison

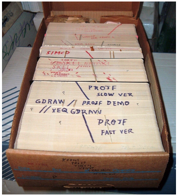
Fortran: Real Talk

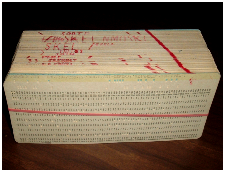
- Fortran has survived **70 years** in the computer science era and is still used today
- Highly powerful compiler design
- Landmark in **code optimization** and **program synthesis**

Fortran: Program representation and storage

- Throughout the 70s, many Fortran programs were stored or directly written on punch cards
- One card was used to store each line (statement) of Fortran program
- A stack of cards was necessary to store an entire program







Fortran: Code Examples

```fortran
  ! Fortran 90 implementation of a monte-carlo algorithm
  ! to approximate pi
program main
  implicit none
  integer :: i, samples, count, report
  real(8) :: x, y, d, estimate

  samples = 10000
  count = 0

  do i = 1, nsamples
     call random_number(x)
     call random_number(y)

     r = sqrt(x**2 + y**2)
     if(r < 1d0) count = count + 1

     if(mod(i, report) == 0) then
        estimate = 4d0 * count/dble(i)
        write(*,*) estimate
  end do
```

Listing: Implementation of a monte-carlo algorithm that approximates π in FORTRAN

Fortran: Code Examples

```fortran
  ! Fortran 90 implementation of the calculation of the euler number
program euler
  implicit none
  integer, parameter :: n = 10
  integer :: s,f
  do i = 1, n
     f = call fact(n)
     s = s + (1 / f)
  end do
end program

integer function fact(n)
  implicit none
  integer :: n,f

  do i = 1, n
     f = f * i
  end do

  fact = f
end function fact
```

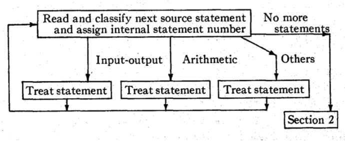
Listing: Implementation of the calculation of the euler number in FORTRAN

An early approach to PS
The Fortran I compiler

- **23,500** assembly language instructions
- Development required **18 person-years**
- Was the first major project in code optimization capable of:
  - Reduction of **runtime**
  - Reduction of **memory occupation**
  - Reduction of **power consumption**

An early approach to PS
The Fortran I compiler

- With the Fortran I compiler various features have been introduced that are used by modern compilers[3]
  - Expression parsing with **redundancy optimization**, **copy propagation** followed by **dead code elimination**
  - Identification of **permutations of operations** → memory access reduction
  - Elimination of **redundant computations** that resulted from common sub-expressions



[3] https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

An early approach to PS

The Fortran I compiler

- Parse and compile arithmetic expressions $\rightarrow$ major challenge
  - Lack of operator priority $\rightarrow$ missing precedence or hierarchy
- The Fortran I compiler expands every operator with parentheses
- For example the expression $\mathbf{A} + \mathbf{B} * \mathbf{C}$ would be expanded to $((\mathbf{A})) + ((\mathbf{B}) * (\mathbf{C}))$
  - Translation scans the resulting expression from left to right
  - Expression according to the operator hierarchical semantics
  - The optimization then eliminates redundancy

Listing: Code after parsing

```
u1=u2+u4;  u2=u3;  u3=A;
u4=u5*u6;  u5=B;  u6=C
```

Listing: Code after optimization

```
u1=A+u4;  u4=B*C
```

An early approach to PS
The Fortran I compiler

a **Dead variable elimination** → Removal of statements assigned to values to unused variables

b **Elimination of common sub-expressions** → search for expressions that have been performed previously

c **Constant propagation** → Removal of calculations that only contain known constants that can be performed directly in the compiler

d **Code motion** → rearrangement of expressions

Listing: sample program

```
A = B + C
Y = Y + i .
Z = C
Q = (Z + B) * SIN (.7854)
DO I=1,100
P(I) = P(I) * (A + B)
```

Listing: optimized program

```
A = B + C
!                          <— a
!                          <— a
Q = A * 0.7071   <— b,c
T = A + B                  <— e
DO I=1,100
P(I) = P(I) * T
```

An early approach to PS

Homework

- Do research on Monte-Carlo algorithms (i.e. Metropolis algorithm)
- Read the papers [Bac+57] and [Sch73]

References

[DS00] Jack Dongarra and Francis Sullivan. "Guest Editors' Introduction: The Top 10 Algorithms". In: *Computing in Science and Engg.* 2.1 (Jan. 2000), pp. 22–23. ISSN: 1521-9615. DOI: 10.1109/MCISE.2000.814652. URL: https://doi.org/10.1109/MCISE.2000.814652.

[Sch73] Paul B. Schneck. "A survey of compiler optimization techniques". In: *Proceedings of the ACM Annual Conference*. ACM '73. Atlanta, Georgia, USA: Association for Computing Machinery, 1973, pp. 106–113. ISBN: 9781450374903. DOI: 10.1145/800192.805690. URL: https://doi.org/10.1145/800192.805690.

[Bac+57] J. W. Backus et al. "The FORTRAN automatic coding system". In: *Papers Presented at the February 26-28, 1957, Western Joint Computer Conference: Techniques for Reliability*. IRE-AIEE-ACM '57 (Western). Los Angeles, California: Association for Computing Machinery, 1957, pp. 188–198. ISBN: 9781450378611. DOI: 10.1145/1455567.1455599. URL: https://doi.org/10.1145/1455567.1455599.