Early Nordic compilers and autocodes

Version 2.0.0 of 2014-09-21

Peter Sestoft

IT University of Copenhagen
Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark
sestoft@itu.dk

Abstract. The early development of compilers for high-level programming languages, and of so-called autocoding systems, is well documented at the international level but not as regards the Nordic countries.
The goal of this paper is to provide a survey of compiler and autocode development in the Nordic countries in the early years, roughly 1953 to 1965, and to relate it to international developments. We also touch on some of the historical societal context.
To appear in Gram, Heide, Impagliazzo (editors): *History of Nordic Computing 2014*. Springer Lecture Notes in Computer Science.

1 Introduction

A compiler translates a high-level, programmer-friendly programming language into the machine code that computer hardware can execute. An "autocode" is a term used in the 1950s to denote a combination of simple compiler, assembler, linker and loader; in general, a program that made loading another program into a computer more convenient than using binary or hexadecimal codes.

Whereas the history of computer hardware and the history of programming languages both are well described, the history of compilers, tying the two other subjects together as it were, is not as thoroughly charted. Nevertheless, surveys by Bauer (1974), Knuth (1977) and Ershov and Shura-Bura (1980) give interesting descriptions of the history of compiler development at the international level. However, they mostly neglect the particular developments in the Nordic countries, especially those before 1960.

In this paper we describe the development of early compilers and autocodes in the Nordic countries in the period 1953–1965. This includes in particular the FA-5 and Alfakod autocodes for BESK in Stockholm; the Mercury Autocode (MAC) for Ferranti Mercury in Oslo; the DASK loader and the Algol compilers for DASK and GIER in Copenhagen; the Algol compilers for various copies of BESK in Sweden; and the Simula language and compiler developed in Oslo.

We describe the features of these compilers and compiler-like systems and how they compare to contemporary international developments such as Brooker's Autocode for the Mercury, Backus's Fortran I compiler for IBM 704, and the Algol compilers by Bauer & Samelson, Dijkstra, Hoare, Randell and others.

We mention, but only to a limited degree, the societal and historical background and motivation for these technological developments.

2 Early international development of compilers

2.1 Surveys and other secondary sources

The earliest survey of autocodes we have found is given by a 1954 MIT MSc thesis by Jones [43]. A brief history of compiler development which focuses mostly on US developments but with few names and dates was given in 1962 by Knuth [46]. A history with much better international coverage and references was given in 1974 by Bauer [6] in his compiler construction book, with Soviet developments described separately by Ershov in an addendum to the second printing [23]. In 1977 Knuth and Pardo surveyed the early development of autocodes and compilers [47] beautifully and quite comprehensively up until the 1957 Fortran I compiler, although with no coverage of Nordic developments.

Naur conducted a survey [53] of completed and ongoing Algol-related compiler development worldwide in January 1962 and reported the results in June [54]. Bromberg in 1963 published a list of almost all existing autocodes and compilers [7] though still missing the Oslo and Stockholm autocodes described in sections 3.3 and 3.4.

2.2 Early Nordic computer hardware

The focus of this paper is the development of compilers, not the development of computer hardware or programming languages, both of which are amply documented and discussed, for instance in the *History of Programming Languages* symposia, the *Annals of the History of Computing*, and several books. Nevertheless, for background, Figure 1 presents some early Nordic computers and their international relatives.

The IAS machine design from the Institute of Advanced Studies in Princeton was described in a series of reports by Goldstine, von Neumann and others 1945–1954. These were quite widely circulated and strongly influenced many other computers even before the IAS machine itself was actually completed. In particular, the IAS machine, BESK, DASK and Manchester Mark I all have 40-bit word length, 20-bit one-address instructions and fixed-point binary arithmetics. The Manchester Mark I was the first computer to have index registers (called B-tubes), and DASK adopted this idea. The Mercury had binary floating-point arithmetics, like several earlier machines developed in the US.

3 Early Nordic autocodes

Figure 2 summarizes, in chronological order, the development of Nordic autocodes and compilers, together with some international developments for context. To give an impression of the various autocodes and compilers, we show how a small computing problem can be solved in each of them.

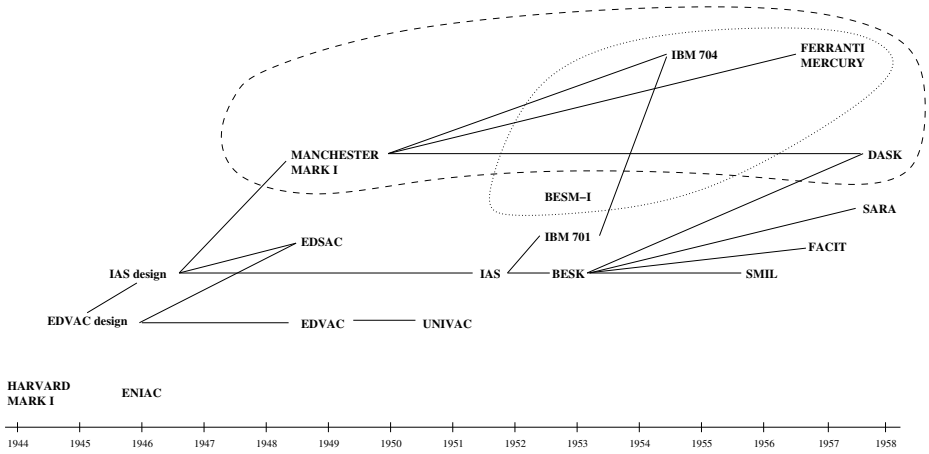

Fig. 1. Genealogy of the early Nordic-made computers BESK and Facit EDB (Stockholm), DASK (Copenhagen), SMIL (Lund) and SARA (Saab in Stockholm), and the British-made Ferranti Mercury (Oslo). The dotted blob encloses computers with binary floating-point arithmetics; the dashed blob encloses computers with index registers.

3.1 An example program

Assume we want to compute the value of this polynomial $p(x)$ of degree 8:

$$p(x) = a_0x^8 + a_1x^7 + \cdots + a_6x^2 + a_7x + a_8$$

If the coefficients a_i are stored in an array `a[i]`, we could compute $p(x)$ using Horner's rule with a C-style for-loop like this:

```
res = 0.0;
for (i = 0; i <= 8; i++)
  res = a[i] + x * res;
```

3.2 The "initial orders" for EDSAC in Cambridge

The so-called "initial orders" for the EDSAC computer at Cambridge University was a set of software routines to load programs in quasi-symbolic form, and link and relocate subroutines, and hence probably constituted the very first autocode; it is included in Bauer's survey [6] but not in Knuth's [47]. It was developed by Wilkes, Wheeler and Gill, was in practical use for years from September 1950, and was the subject of the very first practical book on programming [75] in 1951.

The design was clever and pragmatic. The EDSAC computer read programs from a 5-channel paper tape (as used in telex machines), and Wilkes had arranged the instruction codes so that the character code for an "A" on the tape was also the EDSAC instruction code for addition, "S" also the code for subtraction, and so on. Hence the programmer could work with mnemonic instruction

Source	Machine	Ready	Developer	Ctry	Reference	Comment
	EDSAC	Sep 1950	Wilkes	UK	[75]	"Initial orders"
A-2	Univac I	Nov 1953	Hopper	US	[43, 47]	
Autocode	Man. Mark I	Dec 1955	Brooker	UK	[8, 9, 47]	
IT	IBM 650	Oct 1956	Perlis	US	[7, 46]	
FA-4	BESK	1956	Hellström	SE	[19, 32]	Symbolic addresses
Fortran I	IBM 704	Jun 1957	Backus	US	[4]	Highly optimizing
Alfakod	BESK	Nov 1957	Riesel	SE	[19, 49, 70]	Indexing
MAC	Ferr. Mercury	Dec 1957	Dahl	NO	[12, 37]	Infix, indexing
Runcible	IBM 650	Dec 1958	Knuth	US	[45]	Extends Perlis's IT
Algol 58	Zuse Z22	Dec 1958	Bauer	DE	[72]	No recursion
Algol 60	X-1	Jun 1960	Dijkstra	NL	[20]	Recursion
Algol 60	CDC 1604	Jun 1961	Irons	US	[39]	Recursion
Algol 60	Zuse Z22	Jul 1961	Bauer	DE	[7]	No recursion
Algol 60	DASK	Aug 1961	Jensen, Naur	DK	[41, 42]	No recursion
Algol 60	Facit EDB	Oct 1961	Dahlstrand	SE	[19]	No recursion
Algol 60	Elliott 503	Feb 1962	Hoare	UK	[33, 34]	Recursive descent
Algol 60	SMIL	May 1962	Ekman	SE	[22]	No recursion
Algol 60	GIER	Sep 1962	Jensen, Naur	DK	[55, 56]	Recursion
Algol 60	KDF9	Sep 1962	Randell	UK	[67]	Recursion
Simula I	Univac 1107	Jan 1965	Dahl	NO	[16]	Class, object

Fig. 2. Early Nordic autocodes and compilers, and select international ones.

names instead of hexadecimal instruction numbers, say, yet almost no translation was needed. The chief task of the "initial orders" was to interpret various load-time relocation directives on the program tape, thus achieving relocatable subroutine code on a machine without index registers and relative addressing, and to convert decimal constants and addresses into binary.

3.3 Autocodes for BESK in Stockholm

Initially, the BESK computer read programs from 5-channel telex tape, but used only 4 of the channels, so all programs had to be loaded in hexadecimal code (called sedecimal at the time). According to Dahlquist [17, example 5.13] the example problem from Section 3.1 would be programmed for BESK like this:

```
aaa  AAAOO  comment
200  18050  10000 to register AR
201  20407  set address of instruction at 204 to 100
202  00648  0 to register MR, ie. res := 0
203  00863  res * x to AR
204  FFF28  res * x + a[i] to MR
205  20446  i++, by adding 2 to address part of instruction at 204
206  1820B  compute address-part-of-204 minus hex 113
207  2034E  jump to 203 if difference < 0, so not finished
208  00001  MR to AR
209  18431  AR to cell at address 184
```

The first column contains the instruction addresses `aaa`. The second column contains the five hexadecimal digits constituting a 20-bit instruction, where `AAA` is the address part and `OO` is the operation part. All addresses and constants are in hexadecimal. This assumes that the nine coefficients $a[i]$ are stored at hexadecimal memory addresses 100–112, using two adjacent 20-bit cells for each coefficient; that cell 180 holds 10000; that cell 182 holds 11300; that `x` is at address 008; and that the result must be stored in cell 184. The address and comment parts above are included for explanatory purposes only.

The indexing into the coefficient array is performed by the instruction at 204, and the instructions at 201 and 205 modify the address part of that instruction. Because BESK had no index registers, self-modifying code was essential for array accesses. Clearly, working with such code must have been extremely cumbersome. For instance, if the above code were part of a subroutine, then every time it had to be included in a new program one would have to change the instruction addresses. This would involve not only changing the target addresses of jump instructions (such as 207 above), but also changing the address part of any instructions that edit the address of other instructions (such as 201 and 205 above).

The simple FA-4 and FA-5 autocodes, where FA is an acronym for "fiktiva adresser" or "fictitious addresses", were developed by G. Hellström for BESK for exactly for this purpose [19, 32]. The idea was quite simple: on the program tape one could prefix an instruction with a "fictitious address" such as A204, essentially a symbolic label. Then the FA loader program would change any instruction with address part 204 to instead contain the actual address at which the labelled instruction occurs after loading.

Hence (to the best of our belief) the above example code could be written like this in FA-5:

```
     18050
     20407
     00648
A203 00863
A204 FFF28
     20446
     1820B
     2034E
     00001
     18431
     3000C
```

where all the instructions are exactly as before, but the two instructions previously fixed at addresses 203 and 204 have now been given labels A203 and A204. Hence this code can be relocated by FA-5 to any desirable address; no manual adjustments are needed. This is a big improvement, but the code itself is still very low-level: the programmer must express his intentions in hexadecimal instruction codes and addresses.

The purpose of Alfakod, developed by Riesel, Jonason and von Sydow and commercialized through the company Autocode AB in Stockholm, was to improve on this situation [70, 69, 19]. Alfakod required use of all five channels in the telex tape, and hence was not limited to hexadecimal codes. It provided brief symbolic instruction names (in Swedish) such as `NOL` for "set to zero"; symbolic variable names such as `X`, `Y` and `I`; program labels such as `1` below; subscripting operations such as `Y/I` which means `Y[I]` in modern notation; and tests ("villkor") such as `VMX 1,I,8` which says "go to label `1` if `I<8`"; and more.

Hence the polynomial example could be expressed like this in Alfakod, assuming appropriate declarations:

```
     NOL AR
     FIX 0,I
1    MUL X
     ADD Y/I
     ADX 1,I
     VMX 1,I,8
```

According to Dahlstrand [19, page 33], Alfakod did not see widespread use even in Sweden, perhaps because the concept of externally developed commercial software was still very unusual in the second half of the 1950s.

3.4 Dahl's MAC autocode in Oslo

Unlike Sweden and Denmark, Norway did not build its own computer, apart from the small Nusse machine in the early 1950s. Instead, Forsvarets Forskningsinstitutt (Norwegian Defense Research Establishment, NDRE) in Oslo acquired the very first copy of the commercial Ferranti Mercury computer from the UK. It cost almost 1 million Norwegian kroner and was far more powerful than both BESK and DASK, with binary floating-point arithmetics and 8 index registers.

At first, Jan V. Garwick (see Section 5.1) and Ole-Johan Dahl developed a relatively simple loader program and library of standard routines [15]. Apparently there had been hopes that it would be distributed and used at other Mercury installations, but it was not [64].

Then, under the supervision of Garwick, Ole-Johan Dahl in 1958 developed a very advanced autocoding system MAC for that machine [12]. Although the Ferranti Mercury computer was sold commercially, and Dahl's report describing the system was reviewed in the US the same year [37], there is no evidence that MAC was used outside NDRE, and it is not mentioned in a 1961 list of Ferranti computer literature [24]. After NDRE bought a CDC 3600 computer with a Fortran compiler in the fall 1964, MAC saw little use[1].

Dahl's MAC system supported symbolic variable names, real and complex numbers, 1-, 2- and 3-dimensional arrays, declarations, symbolic labels, composite arithmetic expressions with infix operators and parentheses, efficient code

[1] Dag Belsnes, Oslo, personal communication, 20 May 2014.

generation for multidimensional array indexing, including reduction in strength [13], and many more facilities.

The polynomial evaluation code might look like this in MAC:

```
0 -> A
0 -> n1
Un1 + X A -> A    (1
n1 + 1 -> n1
n1 < 9 ? JUMP1
```

Here `A` holds the result, `U` is the array of coefficients, and `n1` an index variable. The notation was meant to resemble mathematics, so `Un1` means the value of `U` at index `n1`, and `X A` is `X` times `A` with an implied multiplication sign. The "`(1`" is a label, and `JUMP1` a goto statement targeting that label. Note that assignment "`->`" is from left to right, as was quite common in the 1950s.

Dahl's MAC was not the only autocode for the Ferranti Mercury. At Manchester University, R. A. Brooker had developed an autocode system for the Manchester Mark I [8, 47] and a more elaborate one for its successor, the Mercury [9, 10]. Neither Brooker nor Dahl refer to each other's systems, but likely have known about them, since Jan V. Garwick at NDRE had good connections at Manchester and even contributed to the design of the Mercury computer [35, 62].

Indeed there are many similarities between Brooker's and Dahl's systems: they have the same distinction between ordinary variables and index quantities, the same indexing notation `Un1`, the ability to denote multiplication by juxtaposition `X A`, and the support for both real and complex numbers [8]. However, Dahl's MAC is the more powerful one, with more general arithmetic expressions, including parentheses, and declaration of 2D and 3D array variables.

The BESK team in Stockholm knew of Brooker's Autocode around 1955 according to Kjellberg [49, pages 8 and 18].

3.5 DASK in Copenhagen

Peter Naur had used the EDSAC computer in Cambridge and its "initial orders" software extensively February–June 1951 and May–August 1953 for astronomical calculations, and had also visited numerous other computer installations, including BESK in November 1953 [51, 58]. In early 1957, when the DASK computer was being built at Regnecentralen in Copenhagen, he proposed an "external order code" and a similar simple loader routine for it. All of this, including the full (one-page) code of the loader routine, was described in a brief letter to DASK hardware designer Scharøe and project leader Bech [52].

Most of Naur's proposal was adopted, judging from the 1958 DASK programming manual [1], and a comprehensive suite of EDSAC-inspired debugging tools etc. was developed by Jørn Jensen and Per Mondrup [40, 60]. Using the DASK external order code, the polynomial evaluation example might look like this [1, example 5.45]:

```
aaa AAAA I OO  comment
200 2030 A 35  IRB := -18
201 2042 A 44  MR := 0
202    2 B 35  IRB := IRB + 2
203    8 A 0A  AR := x * MR
204  118 B 00  AR := AR + [118+IRB]
205  202 A 33  if IRB<>0 goto 202
```

The general layout of the code is similar to that of the BESK binary code (Section 3.3), but the instruction's address (or immediate) field `AAAA` is now in decimal notation, there is an index register field `I` where `A` means no indexing, `B` indicates index register `IRB`, and so on. The operation code `OO` is still in hexadecimal, but the instruction codes are somewhat simplified from BESK's.

Moreover, the DASK loader routine also supported a form of EDSAC-inspired relative addressing and relocation, not illustrated above, obviating the need for a separate fictitious addressing mechanism [1, chapter 9].

3.6 In retrospect

It seems plausible that several factors saved the team at Regnecentralen in Copenhagen much effort on the development of autocodes and other auxiliary software, so that it could instead focus on the development of compilers for a proper high-level language, Algol 60, as described in Section 4.

First, the Copenhagen team drew much inspiration from the pioneering but pragmatic and simple "initial orders" for EDSAC; second, since the Stockholm developments started much earlier, the Copenhagen team could learn from them; and third, while the DASK computer was based on BESK, it had index registers, more reliable memory hardware (using ferrite cores), and a simplified and cleaned up instruction set, a much more convenient target for code generation.

4 Early Nordic high-level language compilers

Here we give a brief account of Nordic high-level language compilers, chiefly for Algol 60 and Simula, and some of the international context. For broader context, see the surveys by Bauer [6], Bromberg [7], Knuth [47] and Naur [54].

4.1 The Fortran I compiler at IBM

The first compiler in the modern sense for a high-level language is the Fortran I compiler developed by Backus and others at IBM and completed in 1957 [4]. The Fortran I *programming language* had arithmetic expressions with infix operators, precedence and parentheses, like every language today, but otherwise was very simple by modern standards, lacking block structure, nested scopes, procedures, recursion, and so on. Therefore the run-time management of data and programs was nearly trivial, as all variables could be statically allocated at compile-time.

By contrast, the Fortran I *compiler* was remarkably ambitious and advanced. It performed constant folding and common subexpression elimination, optimized index computations (with reduction in strength), and cleverly allocated index registers. The compiler could even perform a Monte Carlo simulation of execution frequencies to optimize the order of branches [4]. Even so, some basic parts, such as the translation of expressions, were implemented in a rather impenetrable manner [74].

The Fortran language is still widely used, and over the years has acquired a large number of features from other languages, including many from Algol.

4.2 International Algol 60 compilers

Unlike Fortran I, the Algol 60 *language* was extremely sophisticated, yet by modern standards a small and elegant language. It had block structure, nested scope, type declarations, recursive functions and procedures, procedures as parameters, and call-by-name as well as call-by-value parameter passing. It did not have user-defined data types, unlike the contemporary COBOL and later Pascal.

Algol provided a large number of hitherto unknown implementation challenges. Bauer and Samelson in 1959 showed how to compile expressions using stacks, much more cleanly than the Fortran I compiler, and also optimize index calculations in a systematic way [71, 72].

A major new challenge in Algol implementation was the run-time allocation and management of data. Dijkstra in 1960 showed how recursive procedures could be executed using a run-time stack of activation records, and how to efficiently access identifiers from statically enclosing scopes using the so-called display [20]. Irons in 1961 created a syntax-directed table-driven compiler, showing that compiler development could be a highly systematic activity [39]. Hoare in 1962 demonstrated how a compiler could be written as a collection of mutually recursive procedures, one for each language construct [33, 34], thereby inventing recursive-descent compilation. Randell and Russell's 1964 book on Algol compiler construction crisply presents many of these techniques [67].

4.3 Nordic Algol 60 compilers

The early development of Nordic compilers in the modern sense is dominated by the Algol 60 compilers developed in Copenhagen, Gothenburg, Lund and Stockholm, and the subsequent development of compilers for Simula in Oslo. In February 1959 Peter Naur from Regnecentralen in Copenhagen became involved in the ongoing international effort to design a common high-level programming language (called International Algebraic Language or Algol 58), and was chosen as one of the seven European members on the Algol 60 committee. He was the only Scandinavian member of the committee, but Ingemar Dahlstrand and others from Sweden, and Jan V. Garwick from Norway, took part in the wider meetings and discussions. The same month Naur also created and became editor of the Algol Bulletin (distributed from Regnecentralen in Copenhagen) as a medium for making and discussing proposals about the language design.

Naur realized that the core problem was to describe this future programming language precisely and clearly, and found that John Backus's recently proposed "Normal Form" was ideal for the purpose. Prior to the January 1960 committee meeting in Paris, he wrote a draft of the entire Algol 60 language description in this style and precise English. This draft was chosen as the basis for the committee's work, and Naur was chosen as editor of the Algol 60 report [5]. There is no doubt that this gave him an intimate knowledge of the language design and of implementation considerations, as well as a strong network of international colleagues, that benefited the subsequent compiler development at Regnecentralen in Copenhagen. These developments, and the somewhat conflicting recollections and interpretations of them, are thoroughly documented elsewhere [59, 65].

The first high-level language compiler in the Nordic countries was the Algol 60 compiler (without recursive procedures) developed by Jørn Jensen, Per Mondrup and Peter Naur for the DASK computer at Regnecentralen in Copenhagen [42]. The same team developed a full Algol 60 compiler for the GIER computer, a transistor-based successor to DASK, also at Regnecentralen [55, 56].

Ingemar Dahlstrand and Sture Laryd developed an Algol 60 compiler (without recursive procedures) for the Facit EDB computer, a clone of BESK, in Gothenburg [18, 19], and later for a Datasaab computer. Dahlstrand's memoirs say that this work drew on papers from Bauer's group in Germany, and on invaluable support and advice from the Copenhagen team [19, page 58].

Torgil Ekman developed an Algol 60 compiler (without recursive procedures) for the SMIL computer, a clone of BESK, in Lund [22]. His supervisor was Carl-Erik Fröberg and his work was furthered by interactions with the Gothenburg and Copenhagen teams [22, page 2].

Börje Langefors at Saab developed the Algol Genius compiler, which supported some COBOL features convenient for administrative data processing [3]. It generated code for for the Datasaab D21 computer, a descendant of BESK, and was based on Dahlstrand's Algol compiler. Later Algol Genius was ported to Univac 1100, based on the Univac Algol compiler from Trondheim [3].

In retrospect, the Nordic Algol implementation efforts do not seem to have had a wider international influence, unlike the work by Bauer, Dijkstra, Irons and Hoare cited above. Even so, the developers of compilers and autocodes faced very nontrivial problems and came up with novel solutions. For instance, the multipass architecture of the Copenhagen Algol compilers provided a clean and efficient way to use the small core memory in combination with the chiefly sequential drum backing store. Similarly, the dynamic two-level virtual memory (core and drum) run-time management of program fragments in GIER Algol [55] seems original and very effective. The 1963 GIER Algol papers [55, 56] described not only technical details but also design rationale, the development team's knowledge sharing, and systematic compiler test. The attention to these aspects paid off: the GIER Algol compiler gained a reputation as highly reliable and usable [73], and the Regnecentralen GIER computer (which sold around 50 copies) came to be seen as an "Algol machine". Low-level machine-oriented debugging tools were not needed and thus never developed for GIER [60].

4.4 The Simula language and compiler

In a highly original development with wide-ranging implications, Ole-Johan Dahl and Kristen Nygaard in Oslo designed the Simula I language and developed a compiler for it on the Univac 1107 computer at the Norwegian Computation Center (Norsk Regnesentral) [16, 63]. The Simula compiler was based on the Univac 1107 Algol compiler, developed at the Case Institute of Technology by Joseph Speroni and others [16, page 678]. The Simula work was the origin of concepts such as class and object, and later strongly influenced the design of widely used languages such as Smalltalk, C++, Java, C#, and many others.

5 Two Nordic digressions

The history of early digital computers is particularly interesting because it coincides with the development of nuclear technology and with the cold war, so exciting side tracks present themselves with every new document discovered.

5.1 The Norwegian nuclear program

Norway's nuclear ambitions are well documented by Forland [25, 26] and their relation to computer science are documented by Holmevik [35, 36] and briefly by Nygaard and Dahl themselves [63, section 1] [14]. Nevertheless we briefly summarize the history here, also because recently declassified material [68] shows the close ties between defense goals and the nuclear programme.

In the summer of 1946, Gunnar Randers and Odd Dahl (no relation to Ole-Johan Dahl) travelled extensively in the United States to study nuclear technology, including the effect of nuclear weapons, after which Randers wrote a frank report to the Norwegian minister of defense [68], declassified in 2013. Randers was head of the newly established Forsvarets Forskningsinstitutt (Norwegian Defense Research Establishment, NDRE) and he and Odd Dahl designed and built an experimental nuclear reactor that became critical in November 1951 [25, 26]. Thus, quite spectacularly, Norway was only the sixth country in the world to have a working reactor.

The computations for the reactor design were led by Jan V. Garwick, as documented in contemporary technical reports [27–30] and in Holmevik's papers [35, 36], and were performed manually and with mechanical calculators. Garwick had succeeded Randers as research assistant to professor of astrophysics Svein Rosseland in 1940 and came to NDRE in 1947. In his reactor computations Garwick was assisted by Kristen Nygaard (at NDRE 1948–1959) who performed Monte Carlo simulations [61]. Later Ole-Johan Dahl, like Nygaard, joined NDRE as a part of his military service and remained there 1952–1961, during which time his MSc thesis [13] and his work on the Mercury Autocode (see Section 3.4) was supervised by Garwick.

From 1962 both Nygaard and Dahl were at the Norwegian Computation Center where they continued to work on simulations and developed the Simula

language, object-oriented programming, and now-ubiquitous concepts such as class and object [36, 63]. Hence object-orientation, like much of early computing, has roots in nuclear technology. Also, it is clear that Jan V. Garwick played a central role in establishing digital computing in Norway; Dahl and Nygaard call him "the father of computer science in Norway" [63, page 245], yet little is written about him, maybe because he moved to the USA in 1967.

6 The origins of the Soviet computer BESM

The origins of the first usable Soviet computer BESM-I, constructed by Sergei A. Lebedev in 1953 [48], are somewhat murky. Some sources, including Nordic ones, claim that it descends from the IAS design or even from BESK (which itself descends from the IAS, see Figure 1).

For instance, an IAS Archives webpage says "*Copies of the IAS machine appeared nationally: [...]; and internationally: BESK in Stockholm; BESM in Moscow; DASK in Denmark; [...] to mention a few*" [38]. It seems[2] that this claim is inspired by Dyson "*The Institute for Advanced Study computer was duplicated, with variation, by [...] BESK in Stockholm, DASK in Denmark, SILLIAC in Sydney, BESM in Moscow [...]*" [21, page 287], but apparently there is no further source for this[3].

Another lead, closer to Sweden, similarly goes nowhere. Annerstedt writes in 1970 "*Besk kom att stå fadder för flera andra datorer: [...] två utländska, den danska Dask och den sovjetiska Besm*" [2, page 104], citing page 82 of the Swedish translation of a book by Peter Naur [57] for this information. However, that page is in an appendix added by the translator, and neither Naur's text nor the appendix mentions BESM or other Soviet computers anywhere at all.

Moreover, whereas the IAS machine, BESK and DASK all have 40-bit word length, 20-bit one-address instructions and fixed-point binary arithmetics, BESM-I has 39-bit word length, 39-bit three-address instructions and floating-point binary arithmetics [48], so it is certainly not a close copy of the IAS design.

Some modern Russian sources in possession of Lebedev's scientific notebooks appear to claim that Lebedev invented everything himself [44]. Although Lebedev clearly was an extraordinarily resourceful designer and constructor, and successful despite difficult circumstances, this seems highly implausible. According to top secret minutes from a closed meeting of the Academy of Sciences in January 1951, Lebedev says "*I have data on 18 machines developed by the Americans. This data has the character of an advertisement, without any kind of information on how the machines are built*" [50, page 5], thus not admitting the availability of any concrete Western designs. But presumably, in 1951, under Stalin, it would be unwise of Lebedev to disclose the Western sources, if any, for Soviet technologies, even in a closed meeting. This is the view also of other biographers of Lebedev [11], and Goodman says "*None of Lebedev's designs was based on close copying of foreign machines*" [31, page 25]. Hence the origins of

[2] Christine Di Bella, IAS Archives, personal communication 18 February 2014.
[3] George Dyson, personal communication 22 February 2014.

BESM-I remain obscure for now, but it is clearly not a direct copy of the IAS machine or BESK.

7 Conclusion and future work

We have given a brief overview of the development 1955–1965 of autocodes and compilers in the Nordic countries, and some of the international and historical context. It is reasonable to conclude that the Nordic autocodes did not attract much international attention, whereas the contributions to Algol 60 language design and description, and the Simula language design, did have considerable international impact.

Later Nordic developments, showing the region's outsize contributions to the area, include Brinch Hansen's Concurrent Pascal and monitors; Hejlsberg's Turbo Pascal, Delphi, C# and TypeScript; the Ada and CHILL compilers from Dansk Datamatik Center; Stroustrup's C++; Augustsson and Johnsson's Lazy ML; Armstrong's Erlang; the Sicstus Prolog and Turbo Prolog/Visual Prolog systems; contributions to Java (by Torgersen and others) and to Standard ML (by Tofte); the virtual machines for Java Hotspot and for Google Chrome by Bak; and much more, both in the industrial and academic spheres.

There is a lot more to say about the general development of compiler technology: lexing and parsing techniques, scope mechanisms, code generation, intermediate languages, type systems and type checking, code optimization, dataflow analysis, run-time organization, abstract machines, garbage collection, linkers and loaders, compiler generators, and much more.

Also, one might investigate how the associated concepts and terminology develops. For instance, who first thought of expressions as trees — abstract syntax — instead of symbol strings? For another example, Backus's Fortran I paper [4] does not use the term "type" in connection with program variables in the 1957, but Perlis's Algol 58 paper [66] in 1958 does; is this the first such use?

Acknowledgements I want to thank all those who helped tracking down and scanning or mailing old documents, who answered my queries about obscure details, or in general discussed the matters here: Christian Gram, Dansk Datahistorisk Forening; Robert Glück, Copenhagen University; Birger Møller-Petersen, University of Oslo; Knut Hegna, University of Oslo Library; Bjørg Asphaug, NDRE Library; Ingemar Dahlstrand and Torgil Ekman, Lund University; Mikhail Bulyonkov, Russian Academy of Sciences, Novosibirsk; Christine di Bella, IAS Archives, Princeton; George Dyson, Bellingham WA, USA; Peter du Rietz, Tekniska Museet, Stockholm; Hans Riesel, Uppsala University; Dag Belsnes, Oslo; Henrik Kragh Sørensen, Aarhus University; and Peter Naur, Copenhagen University. Thanks also to David Christiansen, Olaf Owe and Kasper Østerbye for comments on a draft.

References

1. C. Andersen, N. I. Bech, and O. Møller. *Lærebog i kodning for DASK*. Regnecentralen, Copenhagen, 1958. (In Danish). At http://datamuseum.dk/w/images/d/d3/DASK_kodning.pdf on 17 May 2014.
2. J. Annerstedt. *Datorer och politik*. Zenit, 1970.
3. B. Asker. Algol-genius: An early success for high-level languages. In J. Bubenko, J. Impagliazzo, and A. Sølvberg, editors, *History of Nordic Computing 2003*, volume 173 of *IFIP*, pages 251–260. Springer, 2005.
4. J. W. Backus et al. The FORTRAN automatic coding system. In *Western joint computer conference*, pages 188–198. ACM, 1957.
5. J. W. Backus et al. Report on the algorithmic language ALGOL 60. *Numerische Mathematik*, 2:106–136, 1960.
6. F. L. Bauer. Historical remarks on compiler construction. In F. L. Bauer et al., editors, *Compiler construction, an advanced course*, volume 21 of *Lecture Notes in Computer Science*, pages 603–621. Springer-Verlag, 1974.
7. H. Bromberg. Survey of programming languages and processors. *Communications of the ACM*, 6(3):93–99, 1963.
8. R. A. Brooker. The programming strategy used with the Manchester University Mark I computer. *Proceedings of the IEE - Part B: Radio and Electronic Engineering*, 103(1S):151–157, April 1956.
9. R. A. Brooker. The autocode programs developed for the Manchester University computers. *Computer Journal*, 1(1):15–21, 1958.
10. R. A. Brooker. Further autocode facilities for the Manchester (Mercury) computer. *Computer Journal*, 1(3):124–127, 1958.
11. G. D. Crowe and S. E. Goodman. S. A. Lebedev and the birth of Soviet computing. *Annals of the history of computing*, 16(1):4–24, 1994.
12. O.-J. Dahl. Autocoding for the Ferranti Mercury Computer (MAC). NDRE Report 24, Norwegian Defense Research Establishment, September 1957 1957. iii+92+11 pages. At http://rapporter.ffi.no/rapporter/57/NDRE-rep-24.pdf on 11 May 2014.
13. O.-J. Dahl. Multiple index countings on the Ferranti Mercury computer. NDRE Report 23, Norwegian Defense Research Establishment, August 1957. 1957.
14. O.-J. Dahl. The birth of object orientation: the simula languages. In O. Owe, S. Krogdahl, and T. Lyche, editors, *From object-orientation to formal methods, essays in memory of Ole-Johan Dahl, LNCS 2635*, pages 15–25. Springer, 2004.
15. O.-J. Dahl and J. V. Garwick. *Programmer's handbook for the Ferranti Mercury computer Frederick*. Norwegian Defense Research Establishment, second edition, 1958.
16. O.-J. Dahl and K. Nygaard. Simula, an Algol-based simulation language. *Communications of the ACM*, 9(9):671–678, September 1966.
17. G. Dahlquist, editor. *Kodning för BESK*. Matematikmaskinnämndens arbetsgrupp, Stockholm, 1956. (In Swedish). At http://user.it.uu.se/~foy/Documents/ on 12 May 2014.
18. I. Dahlstrand. The early Nordic software effort. In J. Bubenko, J. Impagliazzo, and A. Sølvberg, editors, *History of Nordic computing, 2003, Trondheim, Norway*, volume 173 of *IFIP*, pages 239–249. Springer, 2005.
19. I. Dahlstrand. *To sort things out. (Ingemar Dahlstrands Minnen)*. Tekniska Museet, Stockholm, 2009. (In Swedish). At http://www.tekniskamuseet.se/ on 13 May 2014.

20. E. W. Dijkstra. Recursive programming. *Numerische Mathematik*, 2:312–318, 1960.
21. G. Dyson. *Turing's cathedral. The origins of the digital universe*. Penguin, 2012.
22. T. Ekman. Konstruktion och struktur av en Algol 60-översättare för datamaskinen SMIL. Technical report, Avdelningen för numerisk analys. Lunds Universitet, May 1962. (In Swedish).
23. A. P. Ershov. Addendum. In F. L. Bauer et al., editors, *Compiler construction, an advanced course. Second edition*, volume 21 of *Lecture Notes in Computer Science*, pages 622–626. Springer-Verlag, 1976.
24. Ferranti Ltd. Classified index of Ferranti computer literature. Technical Report List CS 300, Ferranti, 1961. At At http://bitsavers.informatik.uni-stuttgart.de/pdf/ferranti/FerrantiComputerLiterature_Jul61.pdf.
25. A. Forland. På leiting etter uran: Institutt for atomenergi og internasjonalt samarbeid 1945-51. Forsvarsstudier 3, Forsvarshistorisk Forskningscenter, Norway, 1987. (In Norwegian). At http://brage.bibsys.no/xmlui/handle/11250/99321 on 11 May 2014.
26. A. Forland. Norway's nuclear odyssey: from optimistic proponent to nonproliferator. *The Nonproliferation Review*, 4(2):1–16, 1997. At http://cns.miis.edu/npr/pdfs/forlan42.pdf on 11 May 2014.
27. J. V. Garwick. Beregning av kritisk størrelse for en sfærisk uranmile, delvis omgitt av en reflektor. IR 39, Norwegian Defense Research Establishment, October 1947. (In Norwegian).
28. J. V. Garwick. Beregning av resonansabsorbsjonen i en uranmile hvor uranet er anbragt i parallelle cylindriske staver. IR 40, Norwegian Defense Research Establishment, November 1947. (In Norwegian).
29. J. V. Garwick. Beregning av resonansabsorpsjonen i en uranmile. F 15, Norwegian Defense Research Establishment, January 1951. (In Norwegian).
30. J. V. Garwick. Beregning av temperaturen i de fissile staver i en uranmile. F 49, Norwegian Defense Research Establishment, probably 1951-1952. (In Norwegian).
31. S. Goodman. The origins of digital computing in Europe. *Communications of the ACM*, 46(9):21–25, 2003.
32. G. HellStröm, editor. *Kodning med FA5 för BESK och FACIT EDB*. Matematikmaskinnämndens arbetsgrupp, Stockholm, 1958. (In Swedish). At http://www.tekniskamuseet.se/ on 13 May 2014.
33. C. A. R. Hoare. Report on the Elliott Algol translator. *Computer Journal*, 5(2):127–129, 1962.
34. C. A. R. Hoare. The emperor's old clothes. Turing award lecture. *Communications of the ACM*, 24(2):75–83, 1981.
35. J. R. Holmevik. Compiling Simula: a historical study of technological genesis. *Annals of the history of computing*, 16(4):25–37, 1994.
36. J. R. Holmevik. *Inside Innovation: The History of the SIMULA Programming Languages*. Simula Research Laboratory, Norway, 2005. At http://simula.no/about/history/inside_innovation.pdf on 11 May 2014.
37. J. N. P. Hume. Review of O. J. Dahl: Autocoding for the Ferranti Mercury computer (MAC) [12]. *Mathematical Tables and Other Aids to Computation*, 12(63):259–261, July 1958.
38. Institute for Advanced Study. Electronic computer project. Website. At http://www.ias.edu/people/vonneumann/ecp on 11 May 2014.
39. E. T. Irons. A syntax directed compiler for Algol 60. *Communications of the ACM*, 4(1):51–55, January 1961.

40. J. Jensen, P. Mondrup, and P. Naur. DASK kontrolprogrammer. Technical report, Regnecentralen, Copenhagen, 1959. (In Danish). At http://datamuseum.dk/w/images/2/2a/DASK_kontrolprog.pdf on 13 May 2014.
41. J. Jensen, P. Mondrup, and P. Naur. A storage allocation scheme for Algol 60. *Communications of the ACM*, 4(10):441–445, October 1961.
42. J. Jensen and P. Naur. An implementation of Algol 60 procedures. *BIT*, 1(1):38–47, 1961.
43. J. L. Jones. A survey of automatic coding techniques for digital computers. Master's thesis, MIT, May 1954. At http://dspace.mit.edu/bitstream/handle/1721.1/29846/10903529.pdf?sequence%3D1.
44. V. B. Karpova and L. E. Karpov. History of the creation of BESM: The first computer of S.A. Lebedev institute of precise mechanics and computer engineering. In J. Impagliazzo and E. Proydokov, editors, *Perspectives on Soviet and Russian Computing 2006, Petrozavodsk, Russia*, volume 367 of *IFIP Advances in Information and Communication Technology*, pages 6–19, 2011.
45. D. E. Knuth. Runcible — algebraic translation on a limited computer. *Communications of the ACM*, 2(11):18–21, November 1959.
46. D. E. Knuth. A history of writing compilers. *Computers and Automation*, 11(12):8–18, December 1962.
47. D. E. Knuth and L. T. Pardo. The early development of programming languages. In J. Belzer, A. G. Holzman, and A. Kent, editors, *Encyclopedia of computer science and technology*, volume 7, pages 419–493. Marcel Dekker, 1977. Reprinted in Selected Papers on Computer Languages, CSLI 2003, pages 1-93.
48. S. A. Lebedev. The high-speed electronic calculating machine of the Academy of Sciences of the U.S.S.R. *Journal of the ACM*, 3(3):129–133, 1956. Translated from talk given in Darmstadt, Germany, October 1955.
49. P. Lundin, editor. *Tidig programmering. Transkript av ett vittnesseminarium vid Tekniska museet i Stockholm den 16 mars 2006*. Tekniska Museet, Stockholm, 2007. (In Swedish). At http://www.diva-portal.org/smash/get/diva2:12337/FULLTEXT01.pdf on 18 May 2014.
50. B. Malinovsky. *Pioneers of Soviet computing; edited by Anne Fitzpatrick; translated by Emmanuel Aronie*. SIGCIS, 2010. At http://www.sigcis.org/files/sigcismc2010_001.pdf on 11 May 2014.
51. P. Naur. A view of computers of 1952. (Translated 1997 from Danish original "Beretning om programstyrede elektronregnemaskiner i England, U.S.A., og Sverige"). Unpublished report, May 1954. At http://datamuseum.dk/site_dk/rc/naur/computers1952.pdf on 11 May 2014.
52. P. Naur. Contributions to the design of the dask instruction set, program loader and diagnostics. Letters to Scharøe and Bech at Regnecentralen, January 1957. (In Danish). At http://datamuseum.dk/w/images/6/67/DASK_ordrekode.pdf on 11 May 2014.
53. P. Naur. The questionnaire. *Algol Bulletin*, 14:1–13, January 1962.
54. P. Naur. The replies to the AB14 questionnaire. *Algol Bulletin*, 15:3–51, 1962.
55. P. Naur. The design of the Gier Algol compiler, part I. *BIT*, 3(2):124–140, 1963.
56. P. Naur. The design of the Gier Algol compiler, part II. *BIT*, 3(3):145–166, 1963.
57. P. Naur. *Datamaskinerna och samhället*. Studentlitteratur, 1969. With appendix by the translator Sten Henriksson.
58. P. Naur. Impressions of the early days of programming. *BIT*, 20:414–425, 1980. At http://datamuseum.dk/site_dk/rc/naur/naur3.pdf on 18 May 2014.

59. P. Naur. The European side of the last phase of the development of Algol 60. In R. L. Wexelblat, editor, *History of Programming Languages I*, pages 92–139. ACM, 1981.
60. P. Naur. Personal communication. Letter, June 2014. (In Danish).
61. K. Nygaard. On the solution of integral equations by Monte Carlo methods. NDRE Report F-179, Norwegian Defense Research Establishment, February 1952.
62. K. Nygaard. Jan V. Garwick. In *Norsk biografisk leksikon*. 2009. (In Norwegian). At http://nbl.snl.no/Jan_V_Garwick on 24 May 2014.
63. K. Nygaard and O.-J. Dahl. The development of the Simula languages. *Sigplan Notices*, 13(8):245–272, 1978.
64. T. W. Olle. Ferranti Mercury at the Norwegian Defense Research Establishment. In J. Bubenko, J. Impagliazzo, and A. Sølvberg, editors, *History of Nordic computing, 2003, Trondheim, Norway*, volume 173 of *IFIP*, pages 311–316. Springer, 2005.
65. A. J. Perlis. The American side of the development of Algol. *SIGPLAN Notices*, 13(8):3–14, August 1978.
66. A. J. Perlis and K. Samelson. Preliminary report, international algebraic language. *Communications of the ACM*, 1(12):8–22, December 1958.
67. B. Randell and L. J. Russell. *Algol 60 implementation*. Academic Press, 1964. At http://www.softwarepreservation.org/projects/ALGOL/book/ on 18 May 2014.
68. G. Randers. Rapport til forsvarsministeren. Reise til U.S.A. sommeren 1946 for studium af amerikansk atomforskning. F-RH 2, Norwegian Defense Research Establishment, 1946. (In Norwegian). At http://www.ffi.no/no/Rapporter/46-G.s.-F-RH-2.pdf on 11 May 2014.
69. H. Riesel. Alfakodning, ett lättkodsystem för EDB-maskiner. In *Nordisk symposium över användning af matematikmaskiner, Karlskrona, Sweden*, pages 129–134, May 1959. (In Swedish).
70. H. Riesel, O. Jonason, and L. von Sydow. Alfakodning for matematikmaskiner. allmän del (preliminär version). second printing. Technical report, Alfacode, Stockholm, 1959. (In Swedish, with English abstract).
71. K. Samelson and F. L. Bauer. Sequentielle Formelübersetzung. *Elektronische Rechenanlagen*, 1(4):176–182, 1959.
72. K. Samelson and F. L. Bauer. Sequential formula translation. *Communications of the ACM*, 3(2):76–83, February 1960.
73. N. Sanders. Making computing available. In J. Bubenko, J. Impagliazzo, and A. Sølvberg, editors, *History of Nordic computing, 2003, Trondheim, Norway*, volume 173 of *IFIP*, pages 317–326. Springer, 2005.
74. P. B. Sheridan. The arithmetic translator-compiler of the IBM FORTRAN automatic coding system. *Communications of the ACM*, 2(2):9–21, February 1959.
75. M. V. Wilkes, D. J. Wheeler, and S. Gill. *The preparation of programs for an electronic digital computer, with special reference to the EDSAC and the use of a library of subroutines*. Addison-Wesley, 1951.