LUND UNIVERSITY

Visit to Stanford University, California Aug 1978 - Aug 1979

Elmqvist, Hilding

1979

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Elmqvist, H. (1979). *Visit to Stanford University, California Aug 1978 - Aug 1979*. (Travel Reports TFRT-8024). Department of Automatic Control, Lund Institute of Technology (LTH).

Total number of authors:
1

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

CODEN: LUTFD2/(TFRT-8024)/1-006/(1979)

VISIT TO STANFORD UNIVERSITY, CALIFORNIA
AUG 1978 - AUG 1979

HILDING ELMQVIST

DEPARTMENT OF AUTOMATIC CONTROL
LUND INSTITUTE OF TECHNOLOGY
DECEMBER 1979

Dokumentutgivare
Lund Institute of Technology
Handläggare Dept of Automatic Control
06T0

Dokumentnamn
REPORT

Dokumentbeteckning
LUTFD2/(TFRT-8024)/1-006/(1979)

Utgivningsdatum
December 1979

Ärendebeteckning
06T6

Författare
Hilding Elmqvist

10T4

Dokumenttitel och undertitel
18T0
Visit to Stanford University, California, Aug 1978 - Aug 1979

Referat (sammandrag)
26T0
The author visited Department of Computer Science, Stanford University, California, USA, during the period August 1978 - August 1979. The time was shared between attending courses and seminars, working in a compiler project and own research.

Referat skrivet av
Author

Förslag till ytterligare nyckelord
44T0

Klassifikationssystem och -klass(er)
50T0

Indextermer (ange källa)
52T0

Omfång
66 pages

Övriga bibliografiska uppgifter
56T2

Språk
English

Sekretessuppgifter
60T0

ISSN
60T4

ISBN
60T6

Dokumentet kan erhållas från
62T0
Department of Automatic Control
Lund Institute of Technology
Box 725, S-220 07 Lund 7, Sweden

Mottagarens uppgifter
62T4

Pris
66T0

DOKUMENTDATABLAD enligt SIS 62 10 12

SIS-DB 1

Blankett LU 11:25 1976–07

Report about a visit to Stanford University, California Aug 1978 - Aug 1979

Hilding Elmqvist

1. INTRODUCTION

The author visited Stanford University, California, USA during the period Aug 15, 1978 - Aug 17, 1979. This visit was made possible by financial support from:

Sverige - Amerika Stiftelsen
Department of Automatic Control, Lund Inst. of Tech.
Sixten Gemzeus' fond
Stiftelsen Blanceflor Boncompagni - Ludovisi, fodd Bildt
Stanford University
Ernhold Lundstroms stiftelse
Knut och Alice Wallenbergs stiftelse

I hereby want to acknowledge my thanks for these contributions.

During my thesis work I realized the possibilities with combining knowledge from the field of Automatic Control with Computer Science. A Computer Science department was therefore selected for the visit to USA.

I was formally registered as a postdoctoral visitor at Computer Science Department, Stanford University (Prof Golub). I had an office space at Information Systems Lab. (Prof Kailath).

My main interests were formal languages, language design and compiler technique. The time was shared between attending courses, own research and working on a compiler project. I also participated in Summer Computer Simulation Conference, Toronto, Canada, July 16-18, 1979.

2. COURSES

About one third of the time was spent on courses and seminars. No formal degrees were taken. A list of attended courses is given below.

Autumn quarter

CS 156 Introduction to Mathematical Theory of Computation (Prof Manna)

CS 242 Programming Language Design (Prof Hennessy)

CS 265 Computational Models for the Syntax of Natural Language (Prof Winograd)

Winter quarter

CS 143 Compilers (Prof Hennessy)

CS 246 Operating System Design (Prof Baskett)

EE 385H Seminar in Programming Languages (Prof Hennessy)

EE 392 Program Verification (Prof Luckham)

Spring quarter

CS 224 Models of Thought Processes (Prof Lenat)

CS 243 Compiler Project (Prof Hennessy)

CS 343 Topics in Concurrent Programming (Prof Owicki)

EE 385H Seminar in Programming Languages (Prof Hennessy)

3. RESEARCH

The main part of my research was devoted to features for input and output in programming languages. It was inspired by problems encountered earlier when designing interactive programs. The idea was to have features in programming languages allowing programmers to easily describe the syntax of input and output to the programs. Present languages have only simple features for input - output which means that communication with programs is often inflexible.

It turned out that there is a close relationship between syntactic descriptions of languages and descriptions of how data are stored within computers (cf. the type concept of Pascal).

The basic concepts of grammars are lists of symbols, alternatives among lists of symbols and recursion. The list of symbols corresponds to the list of fields in a record. An alternative is related to the variant part (case) of a

record. Terminal symbols in the grammar correspond to simple data types and to quoted strings. The record definition must be extended to allow quoted strings as well as fields. Meta symbols of the grammar correspond to declared types. Pointer types are needed to make it possible to have recursive definitions. The input and output are obtained by generalized read and write statements.

Below follows an example showing how an extended Pascal notation could be used to define a simple language to enter data about persons.

```
type date = record
  month: (Jan, Feb, Mar, Apr, May, Jun,
          Jul, Aug, Sep, Oct, Nov, Dec);
  day:   1..31;
  ',';
  year: integer;
  end;

person = record
  'name' first: ident; last: ident;
  'born' birth: date;
  case status: (marr, sing) of
    marr: ('married'; mdate: date);
    sing: ('single');
  end;
```

Below follows two examples of inputs that corresponds to the declarations above.

```
name Hilding Elmqvist
born Apr 9, 1948
married Apr 5, 1974


name Bodil Elmqvist
born Mar 19, 1975
single
```

The proposed scheme is documented in

Elmqvist H.: Syntactic Description of Input and Output in Programming Languages (to be published)

A test implementation was done on a Dec-10 computer at Stanford Artificial Intelligence Lab.

4. COMPILER PROJECT

Through Prof Baskett, I became involved in the so called S-1 project. S-1 is a fast computer which has been designed at Stanford. One prototype existed and a simple operating system with Fortran and Pascal compilers was available. A project for development of a better operating system had started. The first step then was to find an appropriate language for systems programming. It was decided to extend Pascal for this purpose. The extensions were defined by Prof Baskett and Prof Hennessy and are documented in

> Hennessy J., Baskett F.: Pascal *: A Pascal Based Sytems Programming Language, Report, Digital Systems Lab, Stanford University

My job was to implement some of these language extensions in an available Pascal compiler. It was done on a Dec-10 computer at Stanford Artificial Intelligence Lab (SAIL).

In an other part of the project this compiler was made to generate machine independent code. An optimizer was also produced. The compiler is programmed in Pascal and can easily be moved to other computers. It is available from Digital Systems Lab, Stanford University.

One of the extensions I implemented was a "parametric type" facility. One often cited drawback of Pascal is the lack of "dynamic arrays". A procedure which, for example, multiplies matrices can only operate on one size of the matrices, say 3 x 5 times 5 x 5. If matrices of different sizes should be multiplied, the procedure needs to be duplicated. This drawback is not present in Algol 60 or Fortran.

One way of overcoming these problems is to introduce parametric types. It is then, for example, possible to declare a parametric matrix like

```
type matrix(n, m: integer) = array[1..n, 1..m] of real;
```

Variables of this type can then be declared as

```
var M1: matrix(3,5); M2: matrix(5,4); M3: matrix(3,4);
```

A procedure mult that multiplies matrices of any size would then have the heading:

```
procedure mult(var Min1, Min2, Mres: matrix);
```

Note that the identifier of the parametric type (matrix) is used without any specific actual type parameters. The actual type parameters can be accessed within the procedure by using a dot-notation (ex. Min1.n, Mres.m). Subranges, records (with variants) and pointers can also be parametric.

The extension to Pascal for parametric types and its implementation are presented in

Hennessy J., Elmqvist H.: The Design and Implementation of Parametric Types (to be published)

5. OTHER ACTIVITIES

I participated in the Summer Computer Simulation Conference, Toronto, Canada, July 16-18, 1979 and presented the paper

Elmqvist H.: Dymola - A Structured Model Language for Large Continuous Systems

This paper presents one part of my Ph.D. thesis. The other part is presented in a paper which I prepared during the stay in California and presented at the IMACS Congress / Simulation of Systems, Sorrento, Italy, September 24-28, 1979:

Elmqvist H.: Manipulation of Continuous Models Based on Equations to Assignment Statements

During my stay in Toronto I also visited David Wortman and David Crowe at Computer Systems Research Lab, University of Toronto. They were implementing a compiler for the programming language Euclid. We discussed what problems they had encountered and compared the implementation schemes we had used for parametric types. There were many similarities.

6. ACKNOWLEDGEMENTS

I want to thank Prof Golub and Prof Feigenbaum for making it possible for me to visit the Computer Science Department at Stanford University. I am also thankful to Prof Hennessy for many stimulating discussions about language design and compiler techniques. Furthermore, I want to thank Prof Kailath for his hospitality.