Turing Award Winner: Frances Elizabeth Allen

V. Krishna Nandivada

PACE Lab
IIT Madras

05-Jan-2018

"If writing programs is an art, then what about efficient programs?"

Sources

1. ACM DL
2. An interview with Frances E. Allen (by Guy Steele, CACM 2011)
3. DBLP
4. Wikipedia

Education

Formal education can only take you so far. After that you are on your own.

- BA (mathematics, New York State College for Teachers, 1954);
- MA (mathematics, University of Michigan, 1957);

Awards

- IBM Corporate Award for Algorithms for Optimizing Compilers (1968)
- Member of the US National Academy of Engineering (1987)
- IBM Fellow (1989)
- IEEE Fellow (1991)
- ACM Fellow (1994)
- Fellow, American Academy of Arts and Sciences (1994)
- IEEE Computer Society Charles Babbage Award (1997)
- Women in Technology International (WITI) Hall of Fame (1997)
- Fellow, Computer History Museum (2000)
- Member, American Philosophical Society (2001)
- Association for Women in Computing Ada Lovelace Award (2002)
- IEEE Computer Society Computer Pioneer Award (2004)
- Anita Borg Technical Leadership Award (2004)
- ACM Turing Award (2006); First woman recipient
- Member of the US National Academy of Sciences (2010)

Publications till Turing Award

Do the bean counting at your own peril.

(As per DBLP)

Publications till Turing Award

Do the bean counting at your own peril.

(As per DBLP)

- Seven conference papers
- Seven journal articles

Contributions of Fran Allen. Zoom level = 0

- Hired by IBM to teach FORTRAN to IBMers.

Contributions of Fran Allen. Zoom level = 0

- Hired by IBM to teach FORTRAN to IBMers.
- The class was unhappy.

Contributions of Fran Allen. Zoom level = 0

- Hired by IBM to teach FORTRAN to IBMers.
- The class was unhappy.
  - the coders in the class could get more performance (manually) than the programs in high-level language!

Contributions of Fran Allen. Zoom level = 0

- Hired by IBM to teach FORTRAN to IBMers.
- The class was unhappy.
  - the coders in the class could get more performance (manually) than the programs in high-level language!
- John Backus (led the FORTRAN project) - had two goals:

Contributions of Fran Allen. Zoom level = 0

- Hired by IBM to teach FORTRAN to IBMers.
- The class was unhappy.
  - the coders in the class could get more performance (manually) than the programs in high-level language!
- John Backus (led the FORTRAN project) - had two goals:
  1. programmer productivity.

Contributions of Fran Allen. Zoom level = 0

- Hired by IBM to teach FORTRAN to IBMers.
- The class was unhappy.
  - the coders in the class could get more performance (manually) than the programs in high-level language!
- John Backus (led the FORTRAN project) - had two goals:
  1. programmer productivity.
  2. application performance.

Contributions of Fran Allen. Zoom level = 0

- Hired by IBM to teach FORTRAN to IBMers.
- The class was unhappy.
  - the coders in the class could get more performance (manually) than the programs in high-level language!
- John Backus (led the FORTRAN project) - had two goals:
  1. programmer productivity.
  2. application performance.
- Fran Allen achieved the second goal with distinction.

Contributions of Fran Allen. Zoom level = 0

- Hired by IBM to teach FORTRAN to IBMers.
- The class was unhappy.
  - the coders in the class could get more performance (manually) than the programs in high-level language!
- John Backus (led the FORTRAN project) - had two goals:
  1. programmer productivity.
  2. application performance.
- Fran Allen achieved the second goal with distinction.
- Her contributions form the basis of most modern compilers.

Contributions of Fran Allen. Zoom level = 1

- Optimizing Compilers

Contributions of Fran Allen. Zoom level = 1

- Optimizing Compilers
- Programming set the tone for her to understand compiling.

Contributions of Fran Allen. Zoom level = 1

- Optimizing Compilers
- Programming set the tone for her to understand compiling.
- Identify structure in the programs.
- Design/implement algorithms to understand programs and their behavior.
- Rewrite the programs to more efficient ones.

Contributions. Zoom level = 2

- Stretch/Harvest computer.
  - Stretch: one of the first supercomputers.

Contributions. Zoom level = 2

- Stretch/Harvest computer.
  - Stretch: one of the first supercomputers.
  - Harvest: A coprocessor for Stretch - for breaking codes.

Contributions. Zoom level = 2

- Stretch/Harvest computer.
  - Stretch: one of the first supercomputers.
  - Harvest: A coprocessor for Stretch - for breaking codes.
  - Built a single compiler framework to handle three languages and two hardware

Contributions. Zoom level = 2

- Stretch/Harvest computer.
  - Stretch: one of the first supercomputers.
  - Harvest: A coprocessor for Stretch - for breaking codes.
  - Built a single compiler framework to handle three languages and two hardware
    1. FORTRAN,

Contributions. Zoom level = 2

- Stretch/Harvest computer.
  - Stretch: one of the first supercomputers.
  - Harvest: A coprocessor for Stretch - for breaking codes.
  - Built a single compiler framework to handle three languages and two hardware
    1. FORTRAN,
    2. Autocoder (similar to COBOL), and

Contributions. Zoom level = 2

- Stretch/Harvest computer.
  - Stretch: one of the first supercomputers.
  - Harvest: A coprocessor for Stretch - for breaking codes.
  - Built a single compiler framework to handle three languages and two hardware
    1. FORTRAN,
    2. Autocoder (similar to COBOL), and
    3. Alpha (a language for breaking codes).

A quick digression: Stretch/Harvest

- Any single Harvest instruction could run for days.
- The instructions could be self-modifying.
- The memory (in 1959) steps were synchronized at clock level (?)
- Total 8/9 systems!

Contributions (contd).

- [1966]: ”Program Optimization”: conceptual basis for systematic analysis and transformation of computer programs.
  - “introduced the use of graph-theoretic structures to encode program content in order to automatically and efficiently derive relationships and identify opportunities for optimization.”

Contributions (contd).

- [1970/71] ”Control Flow Analysis” and ”A Basis for Program Optimization”.
  - established ”intervals” as the context for efficient and effective data flow analysis and optimization.

Contributions (contd).

- [1971] ”A Catalog of Optimizing Transformations”.
  - provided the first description and systematization of optimizing transformations.

Contributions (contd).

- [1973/74] papers on interprocedural data flow analysis
  - extended the analysis to whole programs.

Contributions (contd).

- [1980s] PTRAN project: automatic parallel execution of FORTRAN programs.
  - new parallelism detection schemes.
  - program dependence graph - omni-present in most parallelizing compilers.

Contributions (contd).

- System Y: instead of building the system first and then the compiler, here it was done the other way around.
- Contributions: branch prediction (h/w+s/w), caching, machine independent and dependent optimizations.
- Pre-cursor to the PowerPC system.

Contributions (contd).

- Convincing programmers to use variables for intermediate computations and reuse the same.
  - Making the programmer not worry about where the value is stored.
- Strength reduction; remembering computed values.

Contributions (contd).

- Representing sets as bit-vectors.
- Representing the program as control-flow-graph and propagating sets around the CFG.
- [internal analysis] Identifying loops from the input programs.
- Optimizations on hierarchy of the (control) graphs.

Contributions (contd).

For the PL/I language compiler.

- PL/I has procedures to handle.
- Interprocedural analysis
- Remembering the context.
- How to reuse analysis - say for constant propagation?

PTRAN project

- Goal: improve automatic vectorization.
- Program dependence graphs.
- Identifying useful parallelism - by using a cost model.

Summary I

How do you summarize an elephant? – Focus just on the trunk.

- Did NOT invent new programming languages or language features
- Focus: take programs as they are written, or as programmers like to write them, and making them run really effectively on target machines.

Summary II

How do you summarize an elephant? – or, Focus just on the legs.

- Do many kinds of sophisticated analysis and optimization of the code and to find out as much as you can about the characteristics of the program without actually running it.
  - Static techniques - could be complex + powerful.

Summary III

How do you summarize an elephant? – or, Focus just on the body.

- Using graphs as a representation medium for the program and using a strategy of propagating information around the graph.
  - more than one way exists.
  - make it efficient by using bit-vectors
  - interval analysis provides a mechanism to get an effective order to process the graph nodes.
- Proposed approaches actually work on code and take real programs that aren't artificial benchmarks and make them run.

Summary (summary)

How do you summarize an elephant?

- Two words summary: Fundamental+Impactful.

Summary (summary)

How do you summarize an elephant? – or, just say “elephant”.

- Two words summary: Fundamental+Impactful.