Laudatio:
Neil D. Jones

Fritz Henglein, Jakob Grue Simonsen
DIKU
Starts and False Starts

• From Southern Illinois to Western Ontario

• Work on expressivity of formal languages and machines.
Sojourns, intellectual and otherwise

• Penn State and Kansas

• More expressivity: Of logic. Of control structures.
Kansas and beyond

• Muchnik, Jones, “Even simple programs are hard to analyze” (POPL 1975; J.ACM 1977)

• Jones, Muchnik, “Binding Time Optimization in Programming Languages: Some Thoughts Toward the Design of an Ideal Language” (POPL 1976)

• Jones, Muchnik, “The Complexity of Finite Memory Programs with Recursion” (J.ACM 1978)

• Jones, Muchnik, “Flow Analysis and Optimization of Lisp-Like Structures” (POPL 1979)
Denmark

- 1976-77: Aarhus University (guest prof.)
- 1979-81: Aarhus University (assoc. prof.)
- 1982-87: DIKU (assoc. prof.)
- 1987-2007: DIKU (prof.)
- Since 2007: DIKU (prof. emer.)

- Why move from the US to Denmark?
• Rankings?
• $$?
Denmark
Partial Evaluation

- Binding-time analysis
- mix
- mix-mix-mix
- Jones Optimality
Semantics-based Program Manipulation

- Abstract interpretation
- Supercompilation
- Size-change termination
- Compiler correctness
Language & Complexity

- Language-based characterizations of complexity classes:
  - L and P
  - Efficient simulation
Ph.D. Students (DIKU)

... and PhD students & MS students w/ PhDs at other universities
Coming full circle

- Using programming languages, and programming language constructions for studying computability and complexity.

- Language-based characterization of complexity classes.
“Retirement”

- Termination analysis in higher-order programs
- Programming languages inspired by biology
- Distillation and other program transformations
- (And a smidgen of Partial Evaluation, for good measure)
Criticisms ... Not

- Only 2 papers in 1974
- Only 1 co-author each
- No color graphs
- No benchmark figures
- No energy, climate, aging population...
- Impact?

...and finite model theory
Programs as Data

- Programs
  - … as efficient algorithms
  - … as input to efficient algorithms
  - … as output of correct algorithms
- Semantics-based thinking
SIGPLAN
To explore programming language concepts and tools focusing on design, implementation and efficient use.

Programming Languages Achievement Award

Given by ACM SIGPLAN to recognize an individual or individuals who has made a significant and lasting contribution to the field of programming languages. The contribution can be a single event or a life-time of achievement. The award includes a prize of $5,000. The award is presented at SIGPLAN’s PLDI conference the following June.

2014: Neil Jones

Neil Jones is a pioneer of programming-language methods. He introduced control-flow analysis for higher-order programs, binding-time analysis to tame self-applicable partial evaluation, what is known today as “Jones optimality”, and size-change termination analysis. Neil is also noted for bridges he established between programming languages and complexity/computability theory, e.g., characterizing Turing’s Universal Machine as self interpretation, Kleene’s S-m-n theorem as partial evaluation, Kleene’s second recursion theorem as reflection, and the expressive power of typed cons-free functional languages by means of complexity classes. His pioneering work on complexity theory included the development of completeness for P and solving the spectrum problem (the basis of finite model theory). Lastly, Neil is a tireless and inspiring mentor, and he has written several influential textbooks that are testaments to his thesis that programs are data objects and that programming languages are a cornerstone of computer science.