Formal Software Specification

- **Problem:** In order to improve software quality, we need to understand software better.
- **Solution:** Improve understanding of the intended function of the software, and translate that understanding into implementation.
  - **Formal = syntax + semantics + proof theory**
  - Software specification has evolved because it depends on what you mean by software!

---

The State of SW Specification

- **Growing adoption and success in industry**
  - Matra Transport (Paris Metro) – 100,000 lines of B Spec. Language generated 87,000 lines of ADA code, formally proved, with no errors
- **Plenty of room for improvement in:**
  - Abstraction
  - Scope
  - Integration
  - Efficiency

---

“Guarded Commands…”, Dijkstra, 1975

- Presents a calculus for creating correct code from pre- and post-conditions.
- Generated code is composed of guarded commands
- Emphasis on proving results for non-deterministic code.
- Exploration of the potential for “automatic programming”
Analysis

- Pro Forma Abstract: Radical Solution
- Research Question: method of development
- Result: analytical model
- Validation: trivial example

Questions?
Comments?

State of SW Specification

- Turing: Annotated program state
- Floyd, Hoare, Naur: methods to prove consistency between spec and code
- Dijkstra: formal calculus for spec.
- Broader industry interest
- Mature tools

State of SW Abstraction

- 1940
- 1950
- 1960
- 1970
- 1980
- 1990
- 2000

- Macros, Procedures
- Algorithms, data structures
- Structured programming
- Abstract data types
- Objects
- Packages

Source: Mary Shaw, Lecture "Design and Definition of Data Abstractions", January 2005