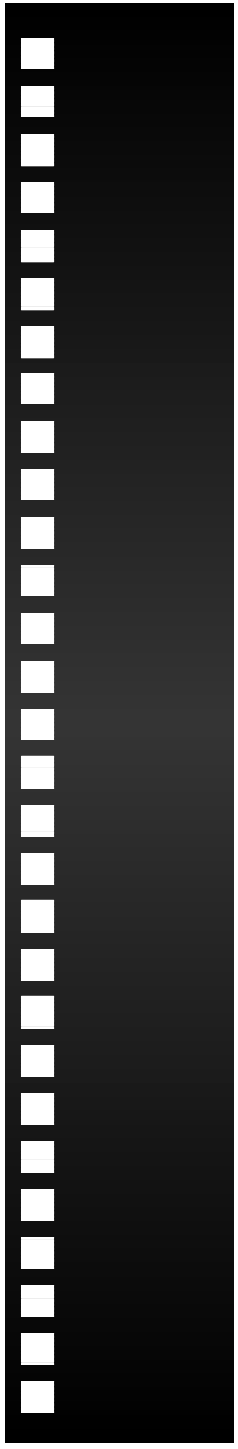
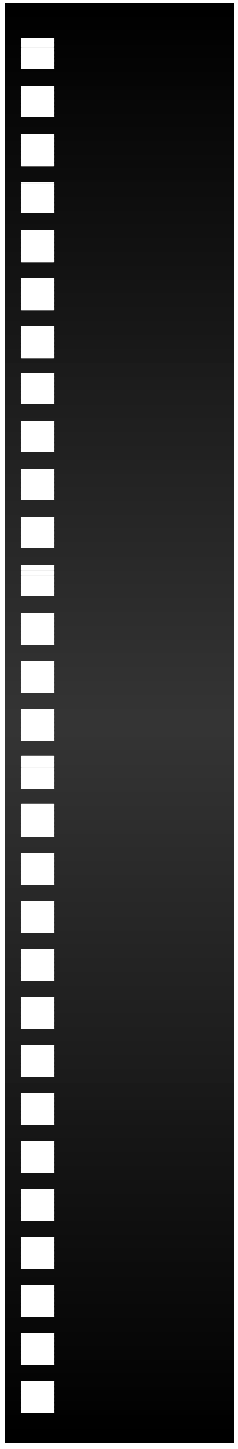


Formal Methods

Michael Collins.

18-849, Section B

Spring 1999



Formal Methods

- Why Formalisms?
- Relationships
- Flaws
- Some systems
- Conclusions

Why Formal Methods?

- We already can build systems.
- Roman Engineers built aqueducts
- Neither group has math for the job
- Both groups waste(d) time & effort

Why Formal Methods? (2)

- Correctness is **proven**, not observed
  - Automatic Proof
- Provides a neutral description
  - Good for documentation
  - Good for standardization
- Legal guarantees

Relationships

- SW Reliability
  - Fault Avoidance
- Fault Tolerant Computing
  - Vide Supra
- Verify/Validate/Certify
  - Serves as a validation system
- (Ultra Dependability)

How do we use them?

- Build a model using a Modeling Language
  - Algebra
- Verify the correctness of the model
  - Theorem Provers exist (Boyer-Moore)
- Translate the model to implementation
  - Again, tools exist

Flaws

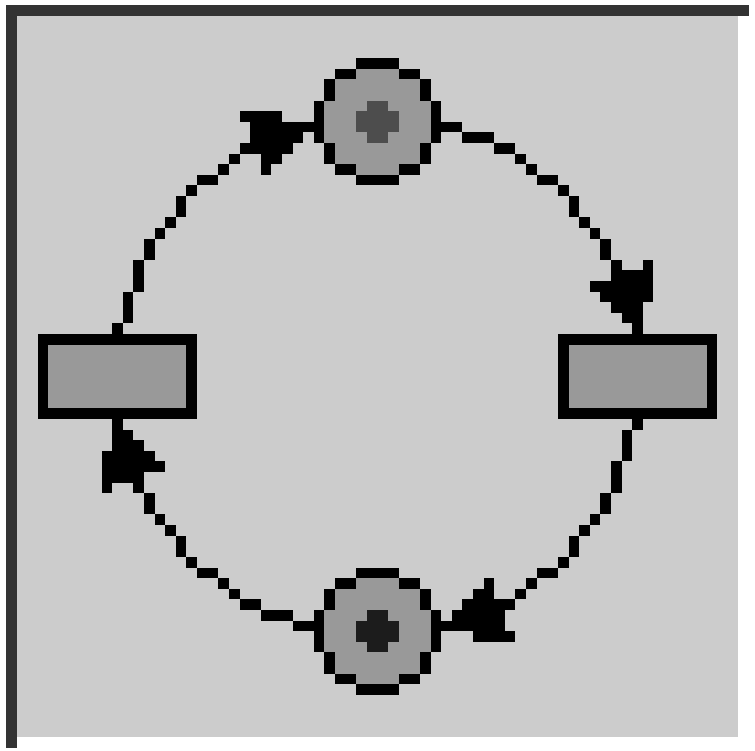
- Idealized models
- Design vs. Implementation
- Learning curve
- How do you prove a prover?
- Can’t apply models to existing systems

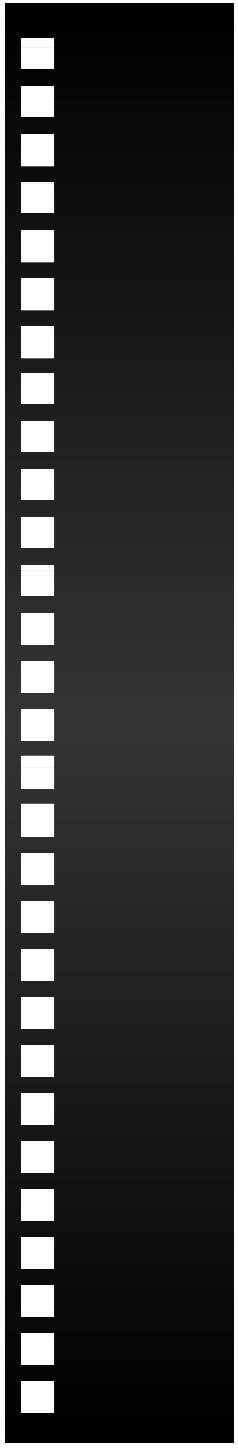
LARCH

- Two-level language
  - Versions for C++, VHDL...
- One language for modeling, one for implementation
- Similar systems include VDM & Z

Petri Nets

- Purely graphical modeling language
- Model Concurrency
- Can be used to define protocols





SML

- Theorem proving language
  - Literally designed for provability
  - Functional and strongly typed
- Proofs are limited in scope
  - No side effects
- Similar projects include Haskell

HOL

- Higher order logic
- Mechanized prover
- Most (in)famous project: Viper Chip
  - Couldn’t handle interrupts
- Other provers include Boyer-Moore

Conclusions

- Formal methods are attractive in theory
- Very few benefits right now
- Current methods provide unsatisfactory models
- Engineers may have to start thinking like mathematicians