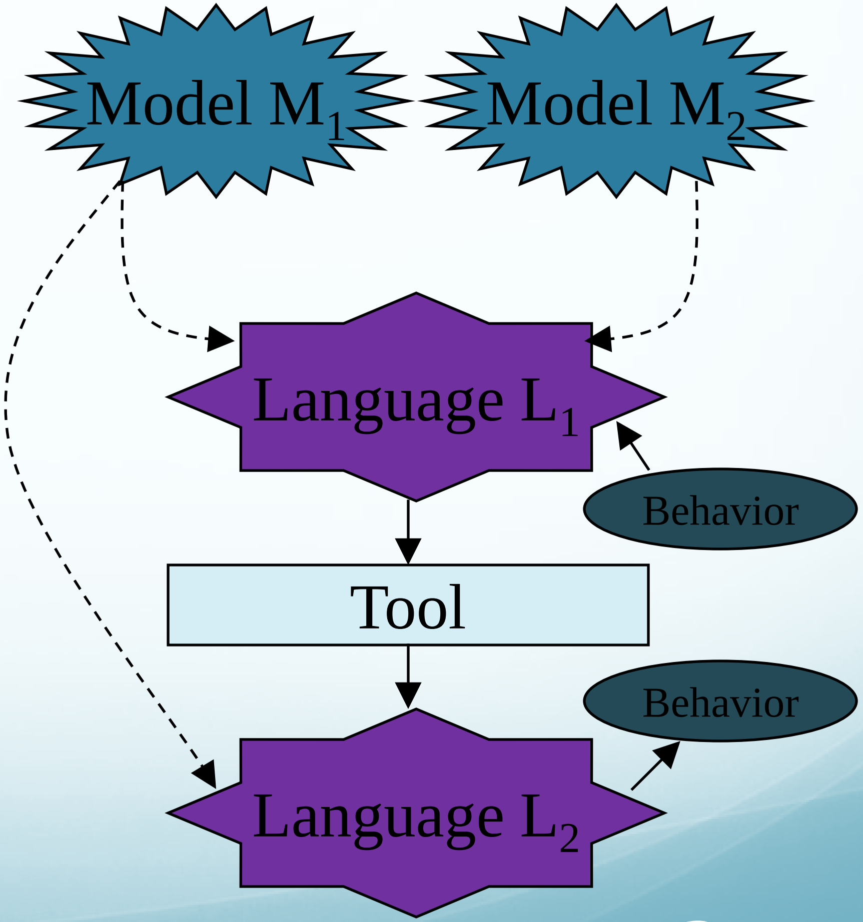
CS245 – Lecture 4

Models, Languages, & Tools

Tony Givargis

Models, Languages, & Tools

- A model of computation is a conceptual notion used to capture system behavior
  - A set of objects
  - Composition rules
  - Execution semantics
- Languages capture models of computation
  - Syntax
  - Semantics
- Tools transform a model captured in one language to a model captured in another language
  - Compilers

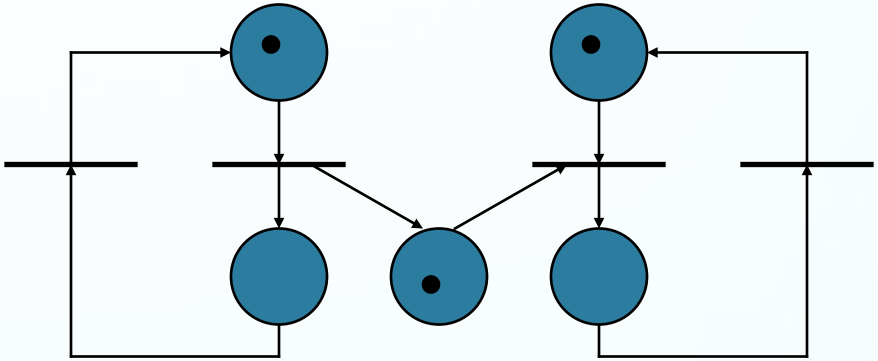


Models of Computation

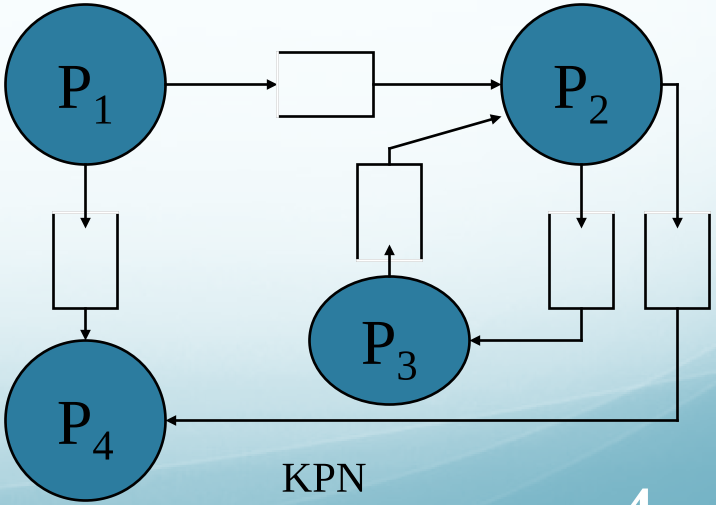
- Sequential model of computation
  - A single thread of execution (all too familiar)
- Concurrent model of computation
  - Multiple threads of execution
  - Synchronization
  - Communication
- Object Oriented (OO) Model of computation
  - View the computation as a set of objects
  - Polymorphism: a derived class can modify the behavior of its base class

Models of Computation

- FSM
  - A set of states and transitions
  - Mealy and Moore
- DFG
  - A set of computation nodes and flow paths
- Petri net
  - A set of places, transitions, edges, and tokens
- Kahn Process Network (KPN)
  - A set of concurrent processes sharing data using unbounded buffers
- Communicating Sequential Processes (CSP)



Petri net



KPN

Meta Models

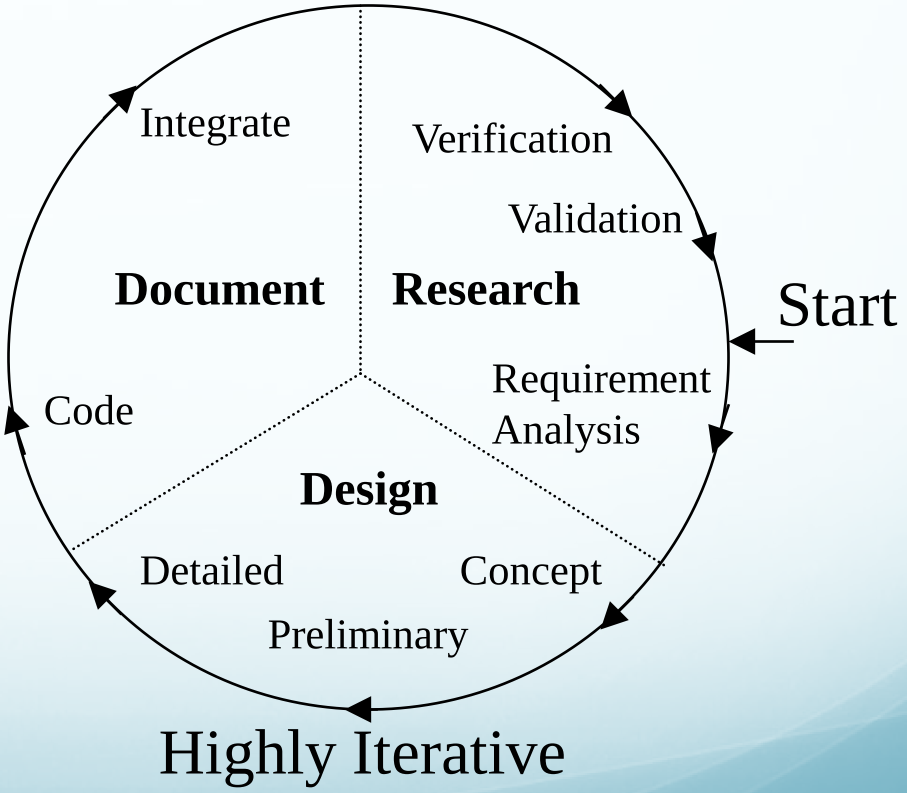
- Discrete Events (DE)
  - Events occur at discrete points on a time continuum
  - Events trigger computations
  - Computations trigger more events
- Continuous Time (CT)
  - Differential equations model continues i/o response as a function of continues time
- Synchronous Reactive (SR)
  - Same as DE
  - All event timings snap to a regular clock
- Publish & Subscribe (P&S)
  - Sending applications (publishers) publish messages without explicitly specifying recipients
  - Receiving applications (subscribers) receive only those messages that the subscriber has registered an interest in
  - Loosely coupled networked systems

Tools

- Compilers
  - Native
  - Cross
  - Optimizing
  - Parallelizing
    - Vectorizing
    - VLIW
  - Special
    - FSM
    - Petri net
  - Synthesis
    - RTL
    - Behavioral
    - System
- IDEs
- Simulators
  - Functional
  - Cycle-accurate
  - Non-functional
  - Bus-functional
- Interpreters
  - Virtual Machine
  - Hypervisor
- Emulators
  - Hardware assisted interpreter
- Debuggers
- Visual programming

Basic Design Flow

- Research
  - Requirement analysis: what is it we are building?
- Design
  - Concept design (e.g., back of an envelop calculations)
  - Preliminary design (e.g., Matlab prototype)
  - Detailed system design
- Document
  - Code
  - Integrate
- Research
  - Verification
  - Validation



Design

- Concept Design
  - Algorithm Design
    - Most interesting systems have a computation core requiring innovative and unique solutions
  - Prototyping at highest level possible
  - Verification and validation of behavior
  - Models play an important role!
- Preliminary Design
  - Functional partitioning: decomposition of algorithm into functional modules
    - Complexity management and early resource allocation
  - Implementation oriented tuning of algorithm
  - System architecture definition
    - Hardware/Software partitioning
  - Models play an important role!
- Detailed design
  - Tools play an important role!

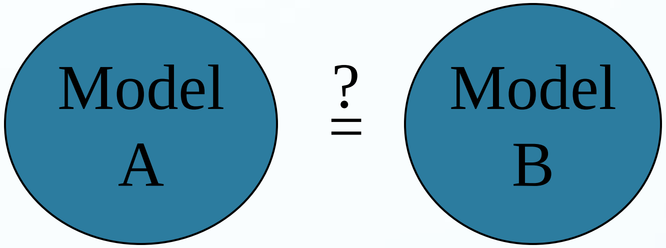
Research

- Verification: *Did we build the thing right*?
  - Simulation
    - Cycle accurate, functional, etc.
    - FPGA prototyping
    - Hardware emulation
    - Product testing
  - Formal
    - See next slide
  - Critical properties are not all functional (user friendliness, security)
- Validation: *Did we build the right thing*?
  - Hard to check (quantitatively)

Abstraction	**Relative-speed**	**Verification Time**
Real-time	1	1 hour
FPGA	10^{-1}	~1 day
Emulator	100^{-1}	~4 days
Behavior (system-level)	1000^{-1}	~1.4 months
Bus functional (system-level)	10000^{-1}	~1.2 years
Cycle accurate (system-level)	100000^{-1}	~12 years
RTL	1000000^{-1}	~1 lifetime
Gate-level	10000000^{-1}	~1 Millennium

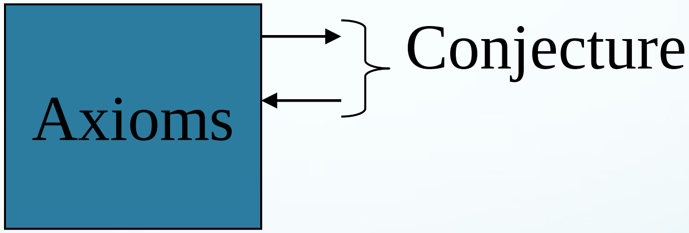
Research: Formal Verification

- Equivalence checking
  - Reduce *A/B* to the canonical form A’/B’
  - Does *A*’ = *B*’ ?



Model A $\overset{?}{=}$ Model B

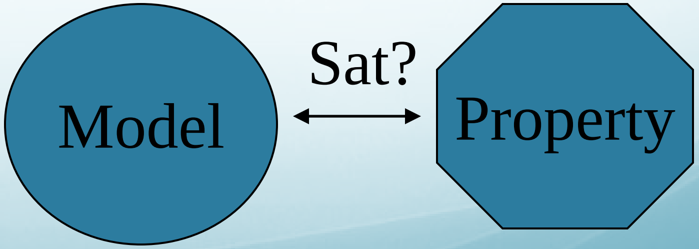
- Theorem proving
  - *Conjecture* (i/o response) is a *logical consequence* of a set of *axioms* (circuit/code)



Axioms

Conjecture

- Model checking
  - *Property* is satisfied by the *model*



Model

Sat?

Property

Document Phase

- Often perceived as the most important phase
    - Certainly important to document
        - To communicate between people, machines, companies, etc.
        - An interface between the research and design phases
        - Languages play an important role