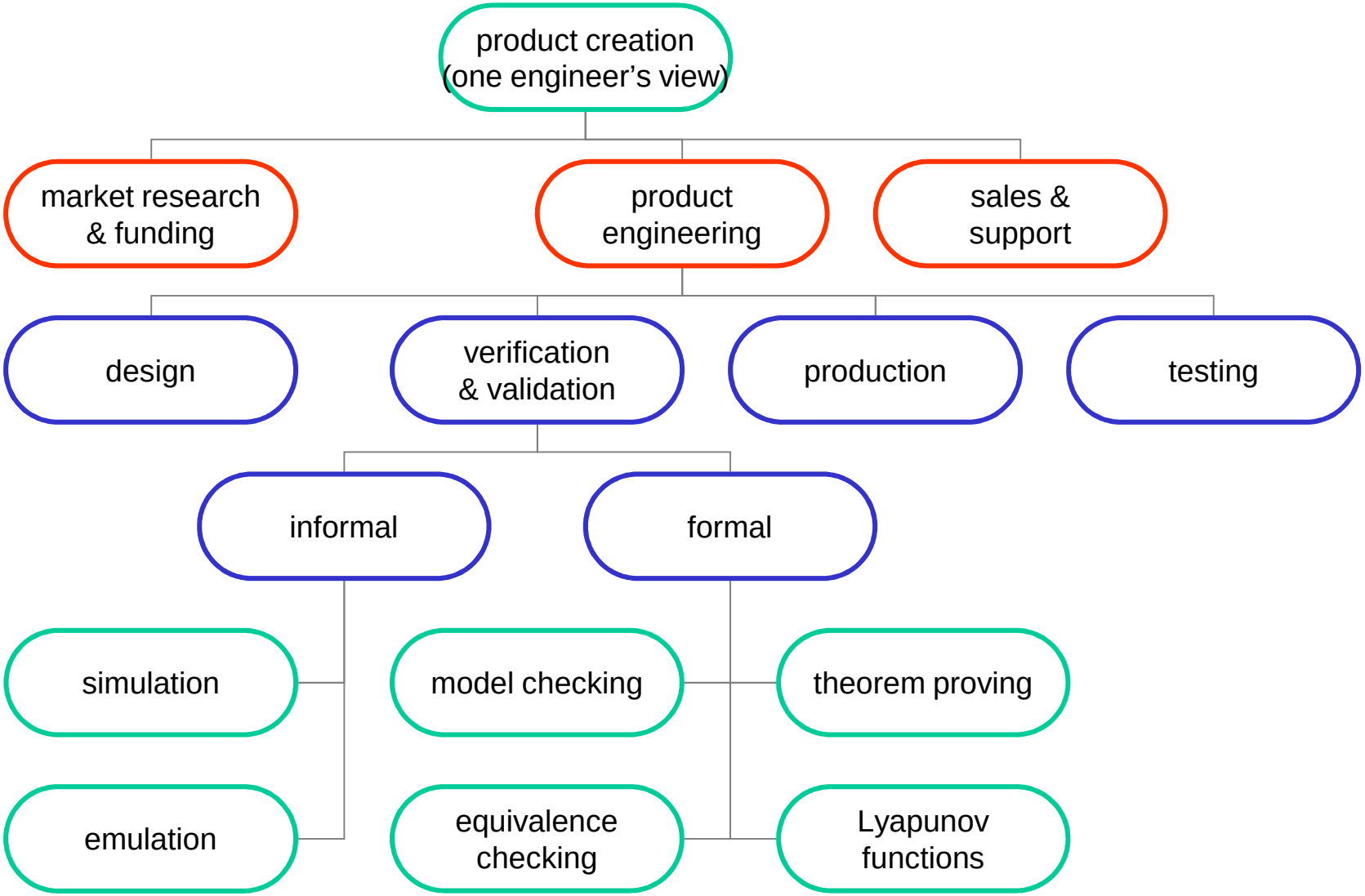
CPSC 513: Integrated Systems Design
Introduction to Formal Verification

Ian Mitchell
Department of Computer Science
The University of British Columbia

On behalf of
Mark Greenstreet & Alan Hu
Integrated Systems Design Lab

Fall Semester, 2009-2010

What is Verification?



Why Use Formal Verification?

- Preproduction verification & validation
  - Physical prototypes are too slow, costly, complex, and/or dangerous to use during iterative design
  - Much cheaper to discover bugs earlier in the design process
- Simulation for early design work
  - User designed test cases can find most bugs
  - Random testing can uncover unexpected bugs
  - Comprehensive input and/or behavior coverage is often impossible
- Verification for (some) late design work
  - Safety critical or high reliability applications must not fail
  - May be easier, cheaper and/or faster to apply formal methods than to design comprehensive tests

Potential Course Topics

- Circuit equivalence with BDDs or SAT
- Dynamic models: finite state automata, temporal logics
- Model checking and reachability
- Theorem proving, SAT Modulo Theories
- Safety / liveness / fairness
- Software verification
- Fixpoint methods: synchronous vs asynchronous models, weakest precondition
- Timed automata
- Hybrid systems: continuous and discrete, timed automata
- Continuous systems: Lyapunov functions, analog circuits
- Models of computation: soundness / completeness / complexity, combining MoCs

Administrivia

- http://www.cs.ubc.ca/~mitchell/Class/CS513.2009W1
- Prerequisites:
  - Graduate standing (CS, math, engineering)
  - Backgrounds vary, so will try to keep course self-contained
  - Be comfortable with logic and proof
- Grades
  - Present a paper or two in class
  - 0 – 3 homework assignments
  - Course project (proposal, oral presentation, written report)
- Collaboration
  - Work together on the problem, but write your own solutions
  - Cite your sources
- References
  - No required text
  - No course notes
  - Many research papers

Conceptual Framework

- Models
  - How do we describe the behavior of the system?
  - Circuits, finite state machines, programs, differential equations, …
- Goals
  - What verification or validation task would we like to accomplish?
  - Equivalence, safety, liveness, fairness, refinement, …
- Techniques
  - What mathematical framework allows us to formally state the problem and determine a solution?
  - Canonical forms, reachable sets, restricted design languages, Lyapunov functions, fixpoint iteration, …
- Tools
  - How do we implement the operations of our technique?
  - Binary decision diagrams, Hamilton-Jacobi PDEs, compilers, …
- Case studies
  - Real problems validated or verified

A Digression: Approaches to Validating Designs

- By construction
  - property is inherent.
- By verification
  - property is provable.
- By simulation
  - check behavior for all inputs.
- By intuition
  - property is true. I just know it is.
- By assertion
  - property is true. Wanna make something of it?
- By intimidation
  - Don't even try to doubt whether it is true

Meret Oppenheim, ***Object***, 1936

It is generally better to be higher in this list