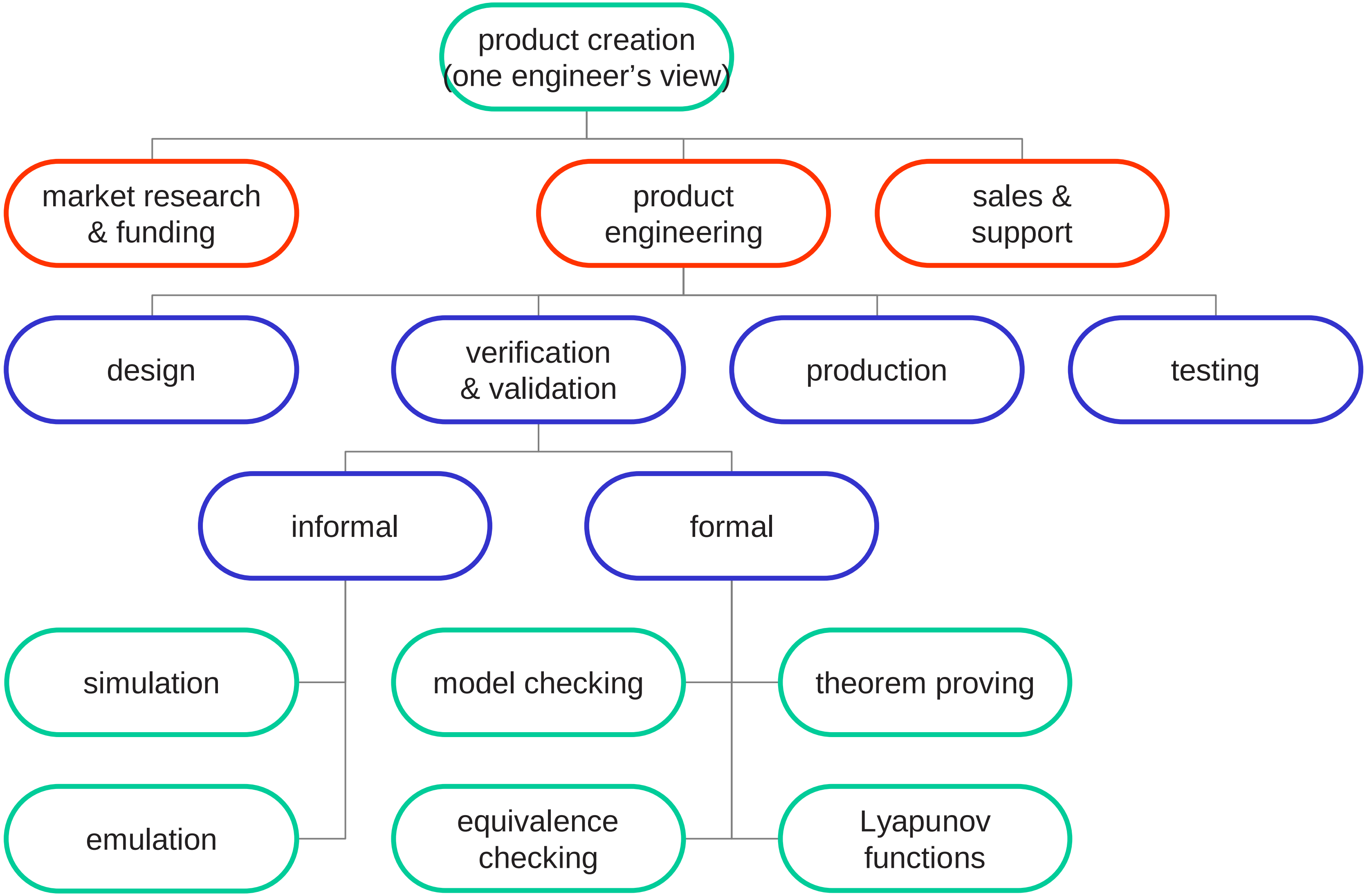
CPSC 513: Integrated Systems Design
Introduction to Formal Verification

Ian Mitchell
Department of Computer Science
The University of British Columbia

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

Fall Semester, 2005-2006

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

Course Topics I

- Introduction
  - Overview: why should you take this course?
  - Administrivia: how do you get a good grade (and hopefully learn something)?
- Circuit equivalence and BDDs
  - Canonical representations
  - Binary Decision Diagrams
- Dynamic models and logics
  - Transition systems, finite state machines & automata
  - Well-posed models, Markovian assumption, nondeterminism
  - Temporal logics: CTL
  - Safety, liveness & fairness
- Model checking
  - Explicit state
  - Symbolic

Course Topics II

- Fixpoint methods
  - Concurrent models: synchronous & asynchronous
  - Weakest precondition
  - Invariants & progress functions
  - Synchronized Transitions
- Timed automata
  - Finite state bisimulation
- Hybrid systems
  - Differential equations for continuous systems
  - Well-posed hybrid models
  - Lyapunov functions
  - Reachability
- Models of computation
  - Soundness, completeness and complexity
  - Moving between MoCs

Administrivia

- http://www.cs.ubc.ca/~mitchell/Class/CS513.2005
- Prerequisites:
  - Graduate standing (CS, math, engineering)
  - Backgrounds vary, so will try to keep course self-contained
  - Be comfortable with logic and proof
- Grades
  - 3 – 5 homework assignments
  - Midterm and/or final exam
  - Mini-project (essentially a project proposal)
- 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, …