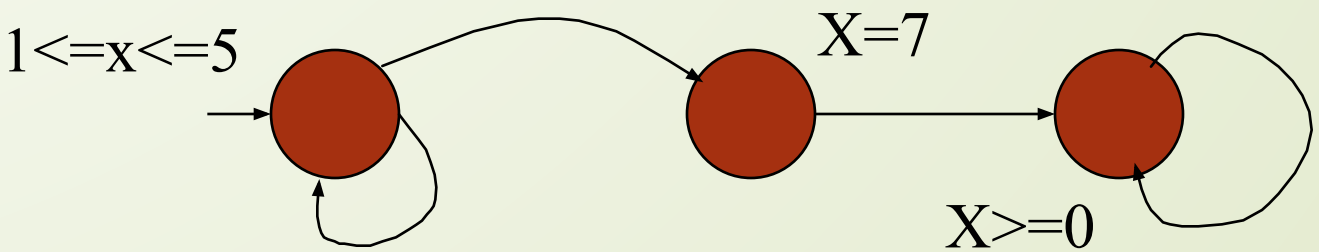
SE4033 Formal Methods for Software Engineering

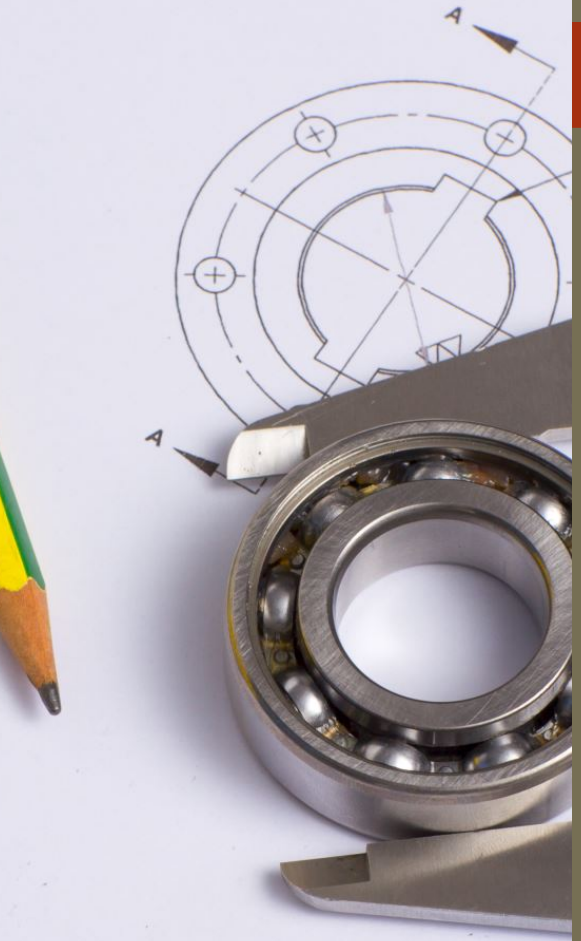
Set of Formal Methods for Software Engineering Phases

Specification: Informal, textual, visual

The value of x will be between 1 and 5, until some point where it will become 7. In any case it will never be negative.

(1<=x<=5 U (x=7/\ [] x>=0))





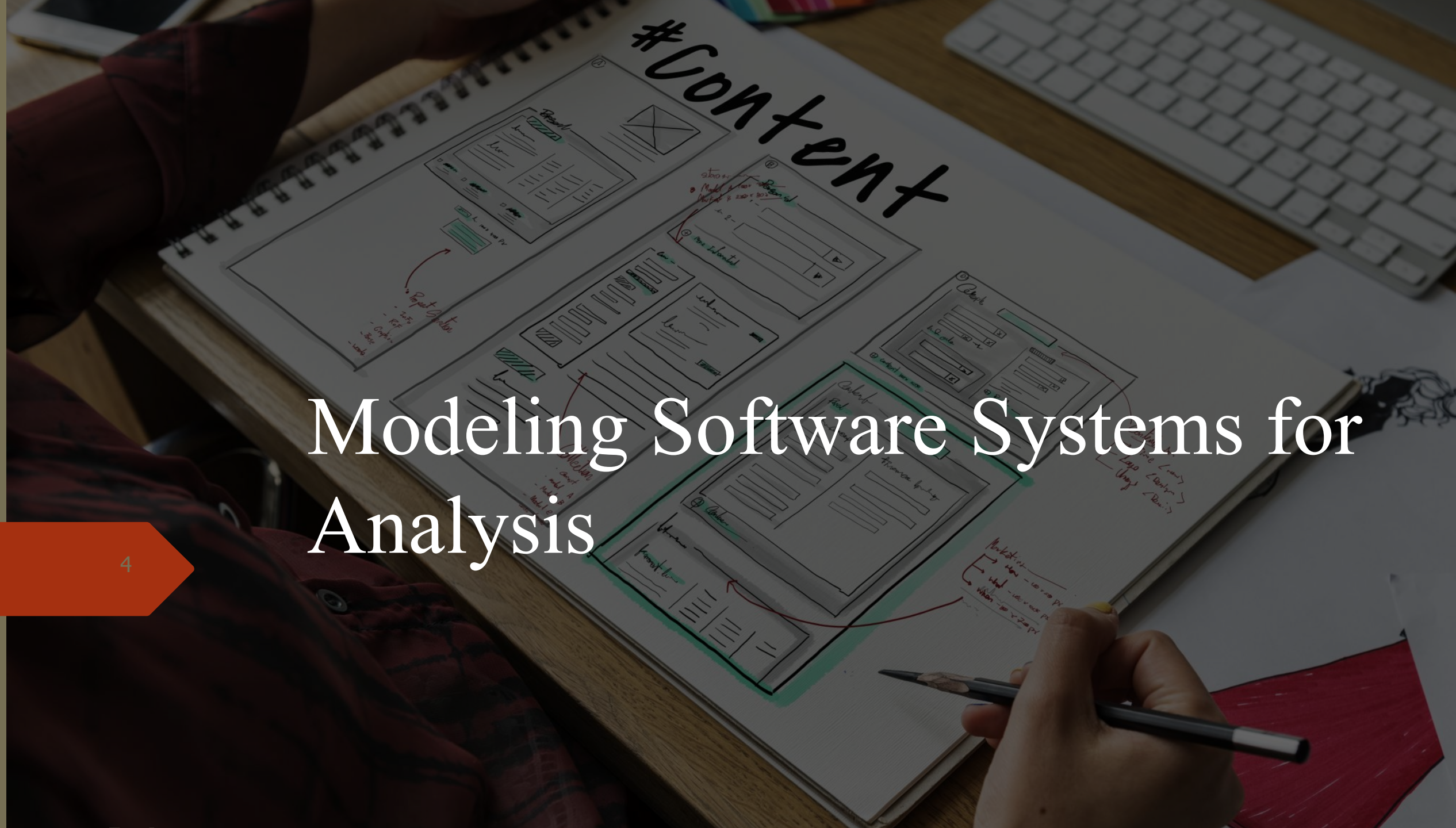
Verification methods

Finite state machines. Apply model checking.

Apply deductive verification (theorem proving).

Program too big, too complicated.
Apply testing techniques.

Apply a combination of the above!



Modeling Software Systems for Analysis

Different types of software

Sequential systems

Concurrent systems (multi-threaded)

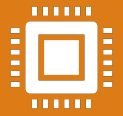
- Distributive systems
- Reactive systems
- Embedded systems (software + hardware)

Protocols

Abstract algorithms

Finite state

Sequential systems



Perform some computational task.

Have some *initial condition*, e.g.,
$\forall 0 \le i \le n$ A[i] integer.

Have some *final assertion*, e.g.,
$\forall 0 \le i \le n-1$ A[i]≤A[i+1].
(What is the problem with this spec?)

Are supposed to terminate.

Concurrent Systems

Involve several computation agents.

Termination may indicate an abnormal event (interrupt, strike).

May exploit diverse computational power.

May involve remote components.

May interact with users (Reactive).

May involve hardware components (Embedded).

A transition system

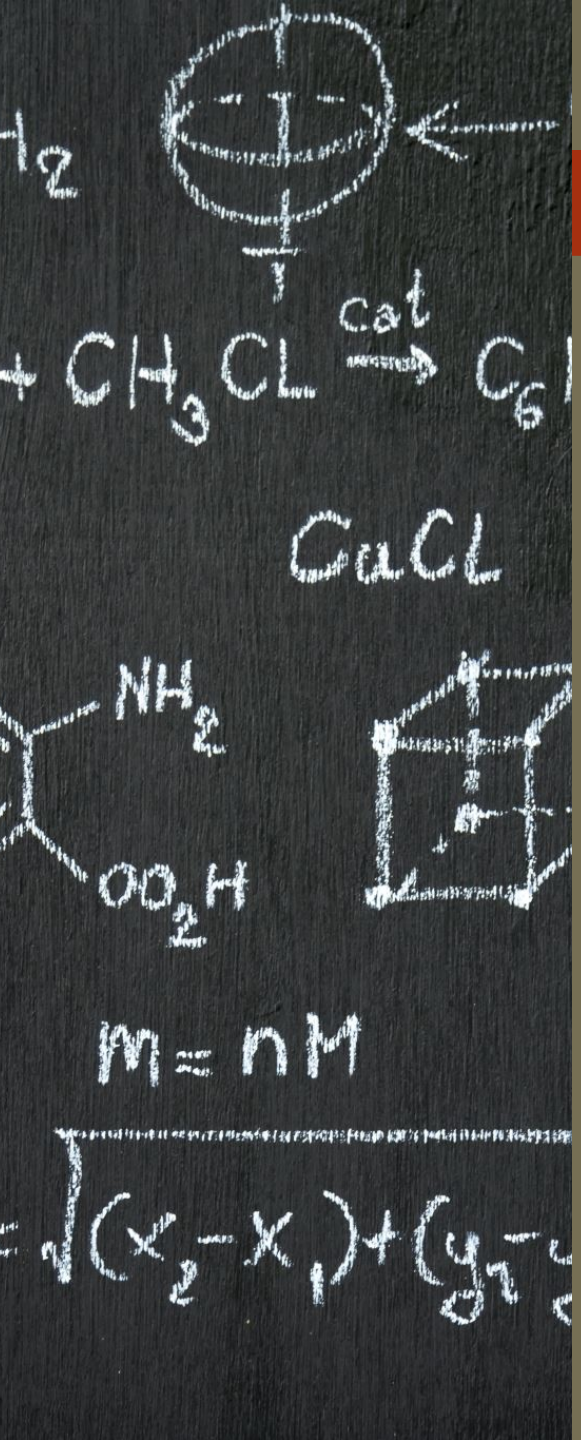
A (finite) set of variables V over some domain.

A set of states S.

A (finite) set of transitions T, each transition $e \square t$ has an enabling condition *e*, and a transformation *t*.

An initial condition *I*.

Example



- V={*a*, *b*, *c*, *d*, *e*}.
- Σ: all assignments of natural numbers for variables in V.
- T={*c* >0□(*c*,*e*):=(*c* -1,*e* +1),
  d >0□(*d*,*e*):=(*d* -1,*e* +1)}
- I: *c* =*a* ∧ *d* =*b* ∧ *e* =0
- What does this transition system do?