CS477 Formal Software Development Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs477

Slides based in part on previous lectures by Mahesh Vishwanathan, and by Gul Agha

April 5, 2013

What is Model Checking?

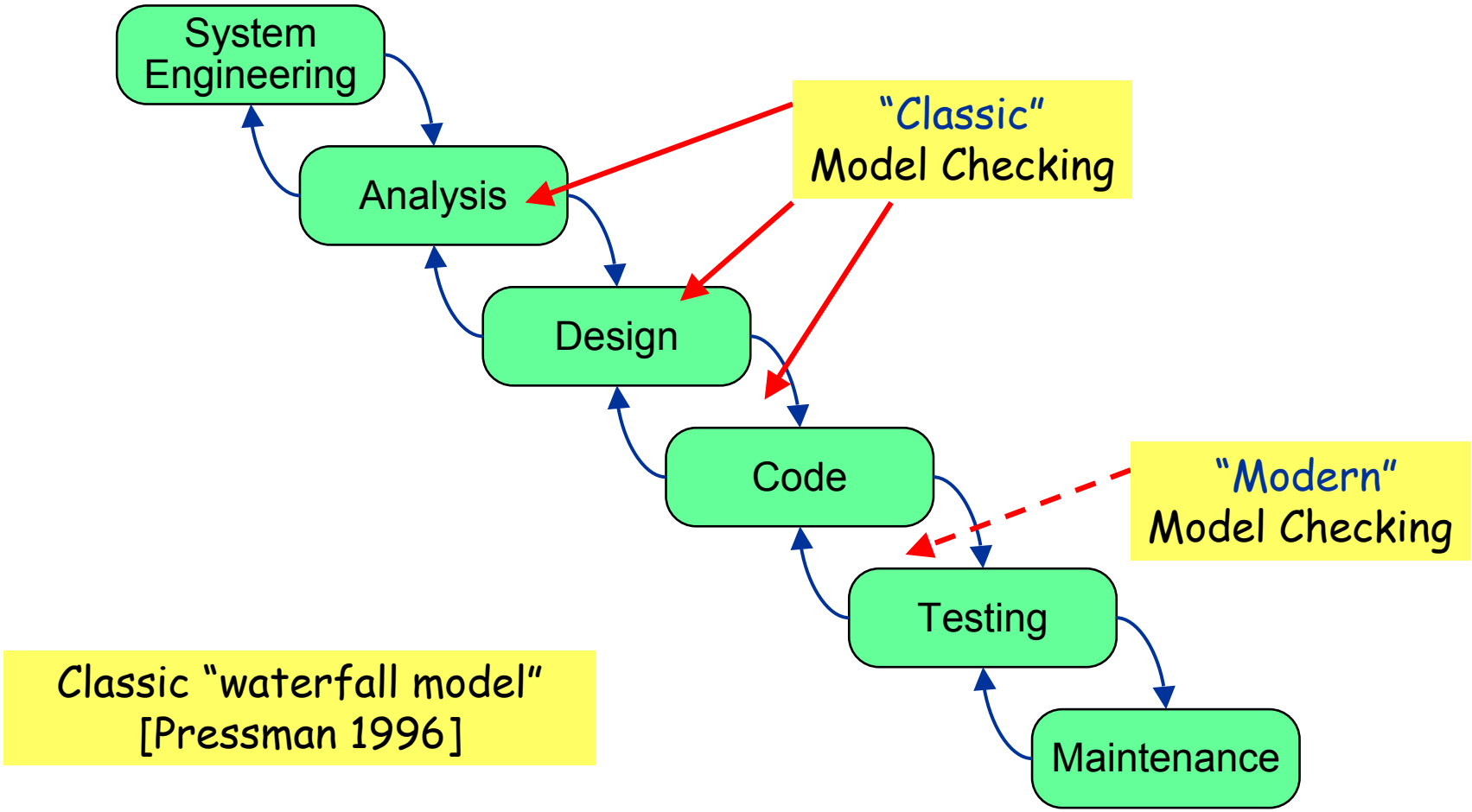
Most generally Model Checking is

- an automated technique, that given
- a finite-state model M of a system
- and a logical property φ,
- checks whether the property holds of model: $M \models \varphi$?

Model Checking

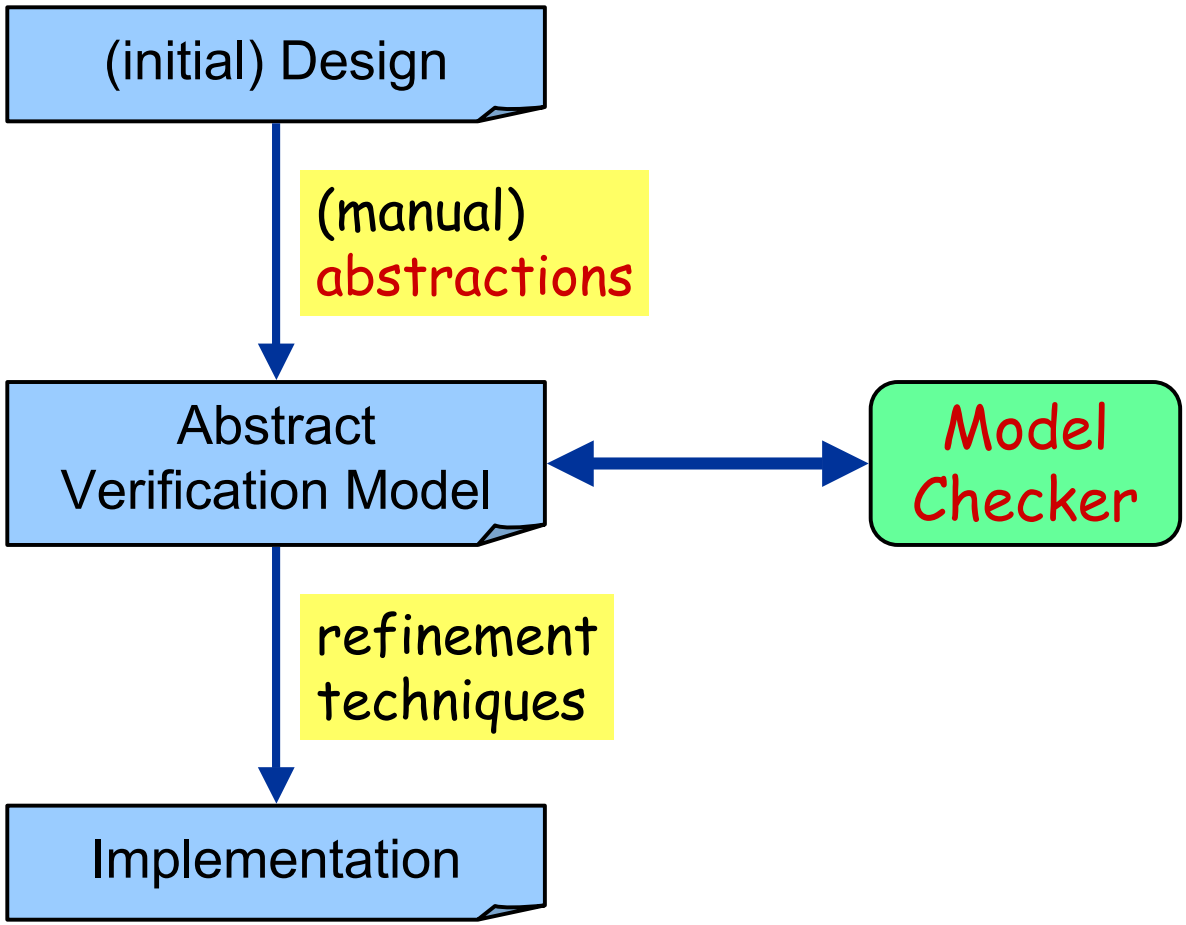
- Model checkers usually give example of failure if $M \not\models \varphi$.
- This makes them useful for debugging.
- Problem: Can only handle finite models: unbounded or continuous data sets can't be directly handled
- Problem: Nnmber of states grows exponentially in the size of the system.
- Answer: Use abstract model of system
- Problem: Relationship of results on abstract model to real system?

System Development



Classic "waterfall model" [Pressman 1996]

"Classic" Model Checking



LTL Model Checking Problem

- Model Checking Problem: Given model $\mathcal{M}$ amd logical property *varphi* of $\mathcal{M}$, does $\mathcal{M} \models \varphi$?
- Given transition system with states Q, transition relation δ and inital state state I, say $(Q, \delta, I) \models \varphi$ for LTL formula φ if every run of (Q, δ, I), σ satisfies $\sigma \models \varphi$.

Theorem

The Model Checking Problem for finite transition systems and LTL formulae is decideable.

- Treat states $q \in Q$ as letters in an alphabet.
- Language of (Q, δ, I), $\mathcal{L}(Q, \delta, I)$ (or L(Q) for short) is set of runs in Q
- Language of φ, $\mathcal{L}\varphi = \{\sigma | \sigma \models \varphi\}$
- Question: $\mathcal{L}(Q) \subseteq \mathcal{L}(\varphi)$?
- Same as: $\mathcal{L}(Q) \cap \mathcal{L}(\neg\varphi) = \emptyset$?

Introduction to SPIN and Promela

- SPIN Background
- Promela processes
- Promela statements
- Promela communication primitives Architecture of (X)Spin
- Some SPIN demo's
  - hello world
  - mutual exclusion
  - alternating bit protocol

Slides from : Theo C. Ruys - SPIN Beginners' Tutorial

SPIN Documentation

- SPIN home page: http://spinroot.com/spin/whatispin.html
- SPIN book: The SPIN Model Checker: Primer and Reference Manual by Gerard J. Holzmann
- On-line Man pages: http://spinroot.com/spin/Man/index.html

SPIN Introduction

SPIN = Simple Promela Interpreter

- Tool for analyzing logical consistenct of concurrent systems
  - specifically data communication protocols
- state-of-the-art model checkers, thousands of users
- Concurrent systems described in modelling language Promela

Promela = Protocol/Process Meta Language

- Resemles C programming language
- Supports dynamic creation of concurrent processes
- limited to describing finite-state systems
- Communication via message channels
  - Synchronous (rendezvous)
  - Asynchronous (buffered)

Promela Models

Promela model consist of:

- type declarations
- channel declarations
- variable declarations
- process declarations
- [`init` process]

A Promela model corresponds with a (usually very large, but) finite transition system, so

- no unbounded data
- no unbounded channels
- no unbounded processes
- no unbounded process creation

Promela Skeleton Example

```
mtype = {MSG, ACK};
chan toS = ...
chan toP = ...
bool flag;

proctype Sender() {
...     /* process body */
}

proctype Receiver() {
...     /* process body */
}

init {
...     /* creates processes */
}
```

Processes

A process type (`proctype`) consists of

- a name
- a list of formal parameters
- local variable declarations
- body consisting a sequence of statements

Sample Process Declaration

```
proctype Sender (chan in; chan out) {
    bit sndB, rcvB;      /* local variables */
    do                   /* body beginning */
    :: out ! MSG, sndB ->
           in ? ACK, rcvB;
           if
           :: sndB == rcvB -> sndB = 1-sndB
           :: else -> skip
           fi
    od                   /* body end */
}
```

The body consist of a sequence of statements.

Processes

A process

- is defined by a proctype definition
- executes concurrently with all other processes, independent of speed of behaviour
- communicate with other processes
  - using global (shared) variables
  - using channels

May be several processes of the same type
Each process has own local state:

- process counter (location within the proctype)
- contents of the local variables

Process Creation

- Processes created with `run` statement
  - Returns process id
- Process createed at any point in exection (of any process)
- Processes start after execution of `run` statement
- Also craeted by `active` keyword before `proctype` declaration

Sample Proctype Declaration Skeleton

```
proctype Foo(byte x) {
    ...
}

active[3] proctype Bar(byte y) {  /* [3] opt; y init to 0 */
    ...
}

init {
  int pid2 = run Foo(2);
  run Bar(17);
  run Foo (27);
}
```