Model Checking @ GSSI

27/3/23

this notes are based on
[1] Baier, Katoen: Principles of model checking. MIT Press (chapters 1-3, 5, and 6)
[2] Clarke, Grumberg, Peled: Model checking. MIT Press

INTRODUCTION

SW is nowaday ubiquitous ⇒ SW correctness is valuable

always relative to specs!

It is fair to state, that in this digital era correct systems for information processing are more valuable than gold.
(H. Barendregt, 'The quest for correctness', in Images of SMC Research 1996

- SW bugs = loss of { lives, money, reputation }

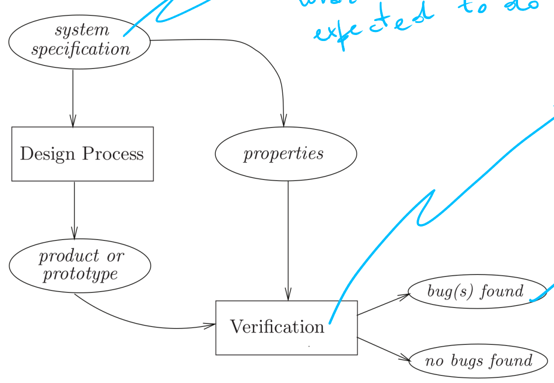
Examples: Therac-25 ≥ 6 deaths between 1985 - 1987
Arianne-5 exploded 36 s after launch
Baggage handling system @ Denver airport (\$ $1.1 \cdot 10^6$ x day x 9 months)
Pentium bug \$ $475 \cdot 10^6$

https://www.reuters.com/business/autos-transportation/us-probing-fatal-tesla-crash-that-killed-pedestrian-2021-09-03
https://www.teslad eaths.com/

- Simulation / testing { + "concrete" artefacts are checked; + "simple"; − partial (when should we stop?) }

- Deductive reasoning { + infinite state systems; − "hard" & time consuming; − interactive }

Borrowed from [1]



what is the system expected to do? AND what is it not expected to do?

Establishes if the system under construction / analysis enjoys some properties

the system either does something not expected or does not do something expected

Exercise. Consider the 3 python functions essentially implementing Example 1.1 in [1]:

```
def inc ():
    while loop :
        if x < bound:
            x += 1

def dec ():
    while loop :
        if x > 0 :
            x -= 1
        loop = x < 0

def reset ():
    while loop :
        if x == bound:
            x = 0
```

Take the temporal property

$$\varphi = \text{Always} \quad 0 \leq x \leq 200$$

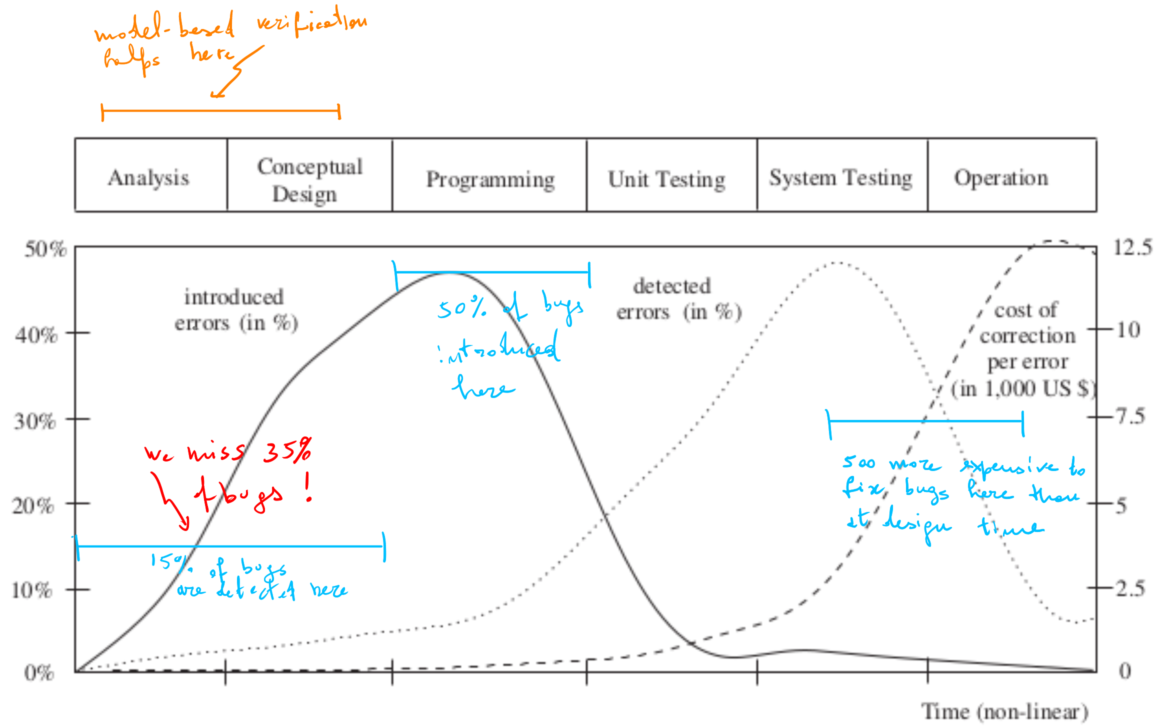
Does φ hold if initially $x == 0$, $loop == True$, $bound == 200$ & inc, dec, and reset above execute concurrently?

We'll focus on MODEL CHECKING, but let's dissect bugs first

Empirical evidence shows that errors do not distribute evenly in
- space (bugs tend to concentrate in few modules) and in
- time

The sooner, the better!

model-based verification helps here



ref.
P. Liggesmeyer and M. Rothfelder and M. Rettelbach and T. Ackermann. Qualitätssicherung Software-basierter technischer Systeme. Informatik Spektrum, 21(5):249–258, 1998.

Quoting [1]
"In software and hardware design of complex systems, more time and effort are spent on verification than on construction. Techniques are sought to reduce and ease the verification efforts while increasing their coverage. Formal methods offer a large potential to obtain an early integration of verification in the design process, to provide more effective verification techniques, and to reduce the verification time."

- Model checking
  - state explosion pb
  - finite state spaces
  - + "easier"
  - + "automatic" (the verification phase +/-)
  - + → "yes" = no bugs c.f. with testing
    → "no" & C.E.

The methodology of MC :=

Behaviour ← sanity check (e.g. simulation)
Properties
Verification ✱
✓

cf p. 3

MODELLING PHASE

design ↦ machine processable models
properties ⤳ typically some (temporal) logics
ideally "push-button", in practice analysis of results

Note
Modelling could be partly "automatic" (eg. compiling from design)
Verification is mainly automatic

Glancing at temporal logics

Note: Temporal logics stem from phylosophy: modal logics to reason about time in natural language!

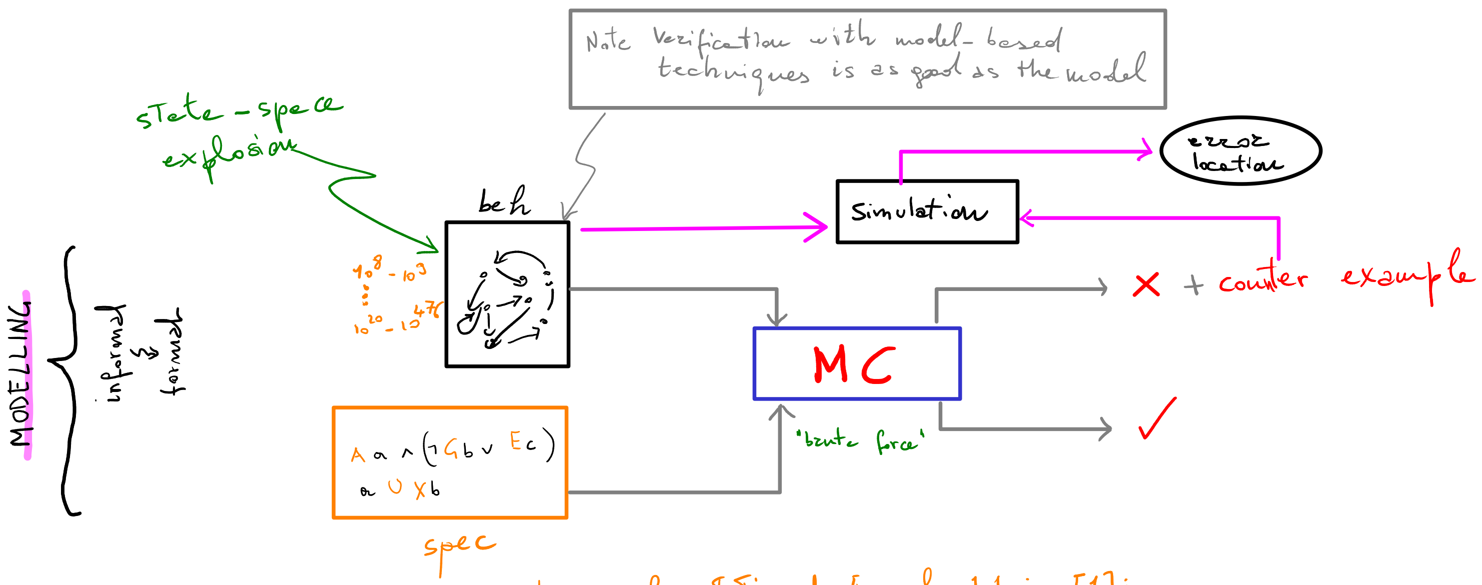
- Designed to predicate on concurrent events
- events are ordered in time
- but time is not explicit

modality $\Box \phi$ eg $\Box(\neg(a \wedge b))$ = it will never happen that events a and b occur "at the same time".

= thread a writes x

= thread b reads x

Schematically cf Fig 1.4 [1]



Exercise. Consider the following python implementation of Example 1.1 in [1]:

```
def inc():
    while loop:
        if x < bound:
            x += 1

def dec():
    while loop:
        if x > 0:
            x -= 1

def reset():
    while loop:
        if x == bound:
            x = 0
```

Does the (temporal) property

Always $0 \leq x \leq 200$

hold if inc, dec, and reset above execute concurrently & initially $x == 0$, loop == True, and bound = 200?

What we're going to see

- Modelling (concurrent) systems
- Temporal logics
  - LTL
  - CTL
  - CTL*
- Fairness conditions
- Hints on symbolic M.C.

if time allows

What we're not looking at

- Partial order reductions
- μ-calculus
- Abstraction techniques
- Quantitative/performance analysis
- Timed models

Modelling

what?

VALIDATION vs **VERIFICATION**

Are we building the right thing?
⇕
is the design faithfully "capturing" the reqs?

Are we building the thing right?
⇕
does the design satisfy the properties?

- Going formal
- Right level of abstraction

≡ precise, but not "cumbersome"

$\boxed{\text{Reactive Systems}}$

(Manna, Pnueli 1995)

- Concurrent
- Interact with an environment ("open")
- possibly non terminating

} NOT FUNCTIONS!

STATE & TRANSITION

snapshot of the system "at a given time"

evolution of the system "in time"

$\boxed{\text{LTS}}$

$$s \xrightarrow{\alpha} s'$$

$\boxed{\text{Kripke Structures}}$

Transition system $TS = (S, Act, \rightarrow, I, AP, L)$

$\boxed{\text{Actions can suitably model interactions, synchronisation \& communication}}$

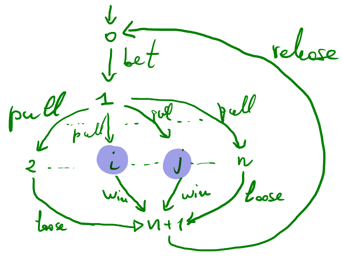
where
- S is a set of states
- Act is a set of actions; in kripke structures Act is a singleton
- $\rightarrow \subseteq S \times Act \times S$ Transition relation
- $I \subseteq S$ are the initial states
- AP is a set of atomic propositions
- $L : S \rightarrow 2^{AP}$

TS is finite if S, Act, and AP are finite
S & $L(s)$ are finite

WLOG we consider transition systems where $I \neq \emptyset$

If $I = \emptyset \Rightarrow$ no behaviour

Example. A (simplified) slot machine



$S = \{0, \dots, n+1\}$ & $I = \{0\}$

$Act = \{bet, win, loose, pull, release\}$

Fix an interval $[i,j]$ with $1 \leq i \leq j \leq n$

we can get rid of $n+1$ and those transitions

$\longrightarrow := \{(0, bet, 1)\} \cup \bigcup_{1 < h \leq n} \left(\{(1, pull, h)\} \cup \{(h, \alpha, n+1) \mid \alpha = \begin{cases} win, & i \leq h \leq j \\ loose, & otw \end{cases}\}\right) \cup \{(n+1, release, 0)\}$

3 wheels

$AP = \{ w_i = f \mid 1 \leq i \leq 3 \ \& \ f \in Fruits\} \cup \{price = n \mid n \in \omega\}$ where $Fruits = \{apple, pear, banana \dots\}$

let $W : \{i, \dots, j\} \longrightarrow Fruit^3$

$L : h \longmapsto \{price = h, \ w_1 = f_1, \ w_2 = f_2, \ w_3 = f_3 \mid w(h) = f_1, f_2, f_3\}$

Exercise: Define L on $h \notin \{i, \dots, j\}$

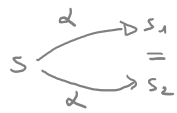
Non-determinism

- crucial modelling mechanism
- under-specification

Deterministic TS $|I| \leq 1$

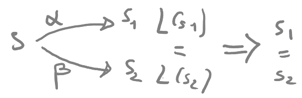
- **action-deterministic** $\forall s \in S, \alpha \in Act : |Post(s, \alpha)| \leq 1$

$Post(s,\alpha) := \{s' \in S : s \xrightarrow{\alpha} s'\}$



- **AP-deterministic** $\forall s \in S \ \forall A \in 2^{AP} : |\{s' \in Post(s) \mid L(s') = A\}| \leq 1$

$Post(s) := \bigcup_{\alpha \in Act} Post(s, \alpha)$



Executions / Traces

Execution fragment $\rho \in$ $\underbrace{S(Act\ S)^*}_{\text{finite}} \cup \underbrace{S(Act\ S)^\omega}_{\text{infinite}}$

s.t. $\rho = s_0 \alpha_1 s_1 \alpha_2 s_2 \dots \alpha_n s_n \dots \Rightarrow s_i \xrightarrow{\alpha_{i+1}} s_{i+1}$ for all i

ρ **maximal** if ρ infinite or
$\rho = s_0 \alpha_1 s_1 \alpha_2 s_2 \dots \alpha_n s_n \wedge Post(s_n) = \emptyset$

ρ **initial** if $s_0 \in I$

Execution Initial maximal execution fragment.

Reachable states $Reach(TS) = \{s \mid \exists \rho$ initial execution fragment ending in $s\}$