Model Checking @ GSSI

Note: experimenting with flipped lectures

9/15/22

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

INTRODUCTION

- Bugs = loss of lives and/or of money

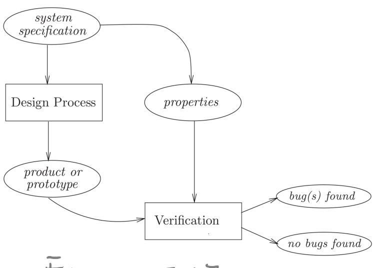
Example Therac-25 ≥6 deaths between 1985-1987
Arianne-5 exploded 36 sec after launch
Pentium bug 485M USD
Baggage handling system @ Denver airport 1.1M USD × day × 9 months

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

- SW ubiquitous ⇒ sw "correctness" valuable → always relative to specs!

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

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



From [1]

- 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 :=

cf p.3

Behaviour ↓ Properties — sanity check (e.g. simulation)

MODELLING PHASE

design ↦ machine processable models
properties ⇝ typically some (temporal) logics

↓

Verification (*)

ideally "push-button", in practice analysis of results

↓ ✓

Note

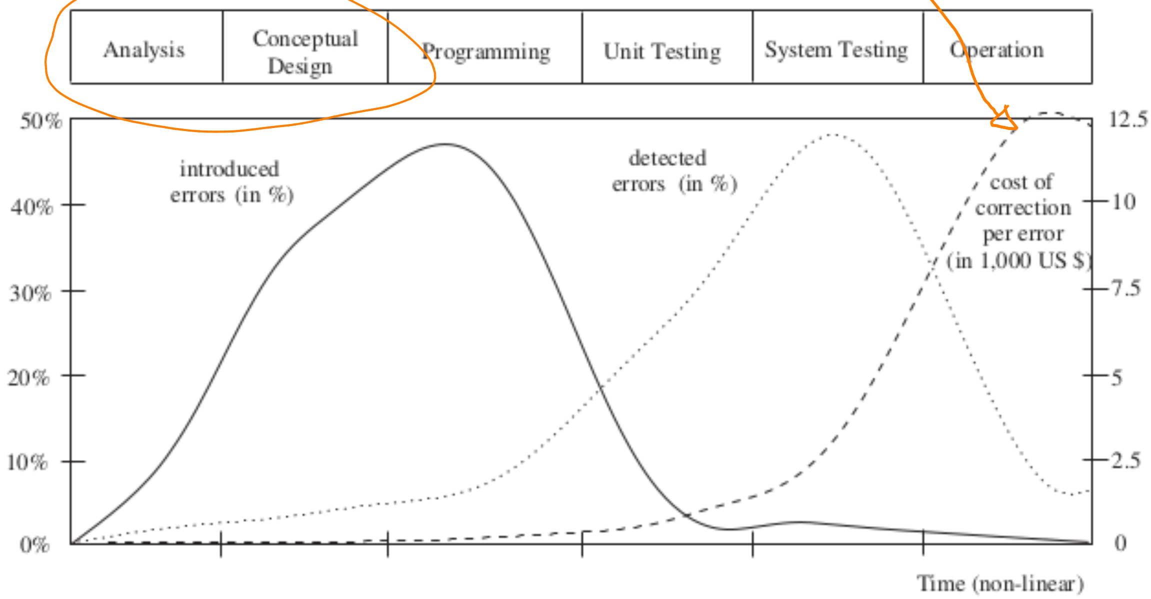
Modelling could be partly "automatic" (eg. compiling from design)
Verification is mainly automatic

(*) Question : If you get an error trace, what do you do?

The sooner, the better!

model-base verification helps here.

Hence, it reduces costs!



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."

Glancing at temporal logics

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

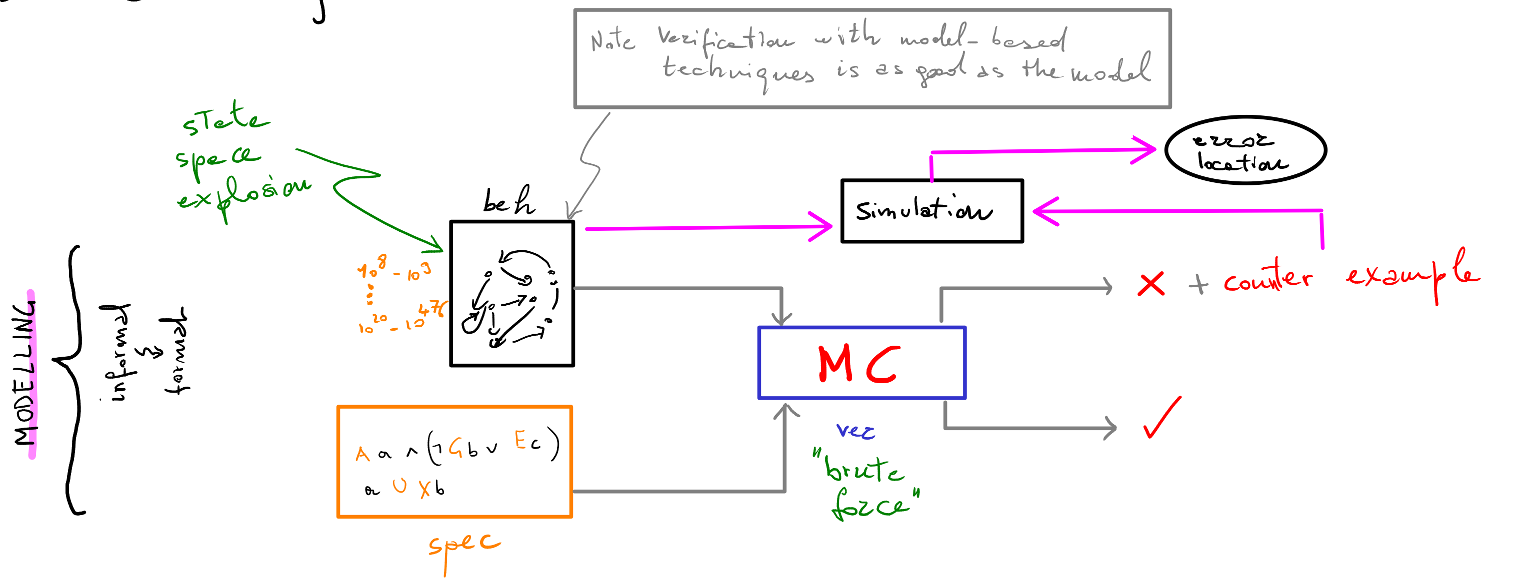
- Designed to predicate on concurrent events
- ordered in time
- but without (explicit) time!

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

a = thread a writes x

b = thread b reads x

Schematically cf Fig 1.4 [1]



Example 1.1 [1] Three parallel processes

def inc(): while true: if $x < 200$: $x := x+1$

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

def reset(): while true: if $x == 200$: $x := 0$

φ = Always $0 \leq x \leq 200$

Exercise: Does φ hold of the parallel execution of the three processes in Example 1.1?

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"

Reactive Systems (Manna, Pnueli 1995)

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

} NOT FUNCTIONS!

STATE — snapshot of the system "at a given time"

&

TRANSITION — evolution of the system "in time"

$\boxed{\text{LTS}}$

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

$\boxed{\text{KRIPKE Structures}}$

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

$\boxed{\text{Actions come handy to model interactions}}$

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

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

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

we can get rid of $n+1$ and those trans.

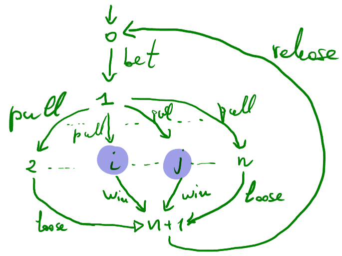
$\rightarrow = \{(0, bet, 1)\} \cup \bigcup_{1 \le h \le n} \{(1, pull, h)\} \cup \{(h, \alpha, n+1) \mid \alpha = \begin{cases} loose & \text{if } \neg w(h) \\ win & \text{if } w(h) \end{cases}\} \cup \{(n+1, release, 0)\}$

3 wheels

$AP = \{ w_i = f \mid 1 \le i \le 3 \text{ \& } f \in Fruits\} \cup \{price = n \mid n \in \omega\}$

where $Fruits = \{apple, pear, banana, \ldots\}$. Let $W = \bigcup_{h \in \{i, \ldots, j\}} \{(h, f_{h_1}, f_{h_2}, f_{h_3})\}$

$L: h \mapsto \{price = h, w_1 = f_1, w_2 = f_2, w_3 = f_3\}$ if $h \in \{i, \ldots, j\} \wedge (h, f_1, f_2, f_3) \in W$



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

Non-determinism

- crucial modelling mechanism
- under-specification

Deterministic TS $|I| \le 1$

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

$:= \{s' \in S : s \xrightarrow{\alpha} s'\}$

$s \xrightarrow{\alpha} s_1$, $s \xrightarrow{\alpha} s_2$, $s_1 = s_2$

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

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

$s \xrightarrow{\alpha} s_1$, $s \xrightarrow{\beta} s_2$, $L(s_1) = L(s_2) \Rightarrow s_1 = s_2$

Executions / Traces

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

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

ρ **maximal** if ρ infinite or

$\rho = s_0 \alpha_1 s_1 \alpha_2 s_2 \cdots \alpha_n s_n \wedge Post(s_n) = \emptyset$

ρ **initial** if $s_0 \in I$

Execution Initial maximal execution fragment.

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