Timed model checking of fault-tolerant nuclear I&C systems

Igor Buzhinsky, Antti Pakonen

igor.buzhinskii@aalto.fi, antti.pakonen@vtt.fi

INDIN 2020 conference

Aalto University
School of Electrical
Engineering

ITMO UNIVERSITY

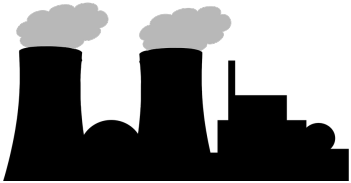
July 21, 2020

Motivation: nuclear power plant I&C verification in Finland

VTT stuk

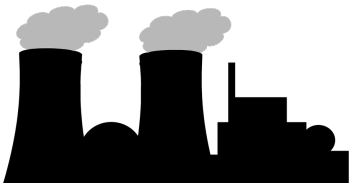
fortum TVO

FENNOVOIMA

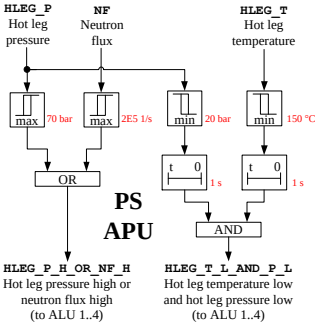


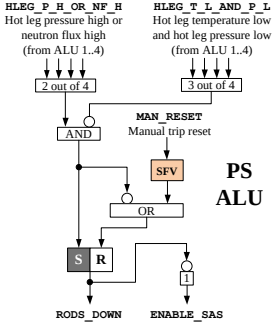
- In Finland, a **formal verification** method called **model checking** is used to ensure the safety of nuclear power plants

Motivation: nuclear power plant I&C verification in Finland



- In Finland, a **formal verification** method called **model checking** is used to ensure the safety of nuclear power plants
- Specifically, VTT model-checks **instrumentation and control (I&C) systems** for Finnish power utilities





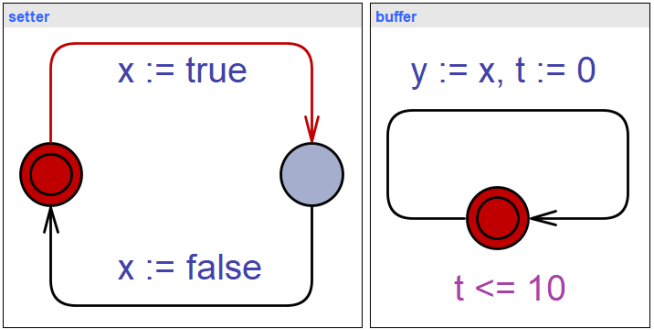
Model checking

- Requires a **formal model** of behavior of the system
- Mathematically, this model is often represented as some form of a state machine
  - For example, UPPAAL allows designing models visually
- There are formal languages that allow specifying formal models textually
  - **NuSMV** and **nuXmv**

Model checking

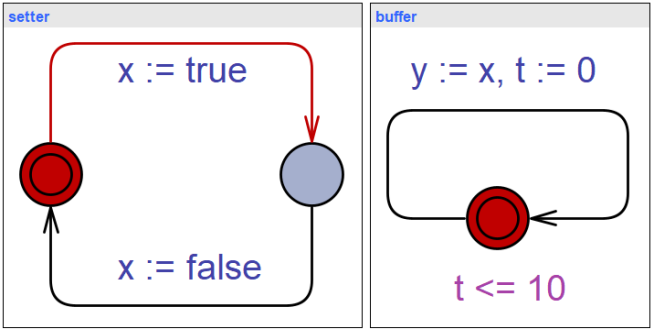
- Requires a **formal model** of behavior of the system
- Mathematically, this model is often represented as some form of a state machine
  - For example, UPPAAL allows designing models visually
- There are formal languages that allow specifying formal models textually
  - **NuSMV** and **nuXmv**
- Similarly, **temporal logics** are formal languages that specify properties to be verified

Formal modeling with timed automata (1)



- Example of a system of two timed automata in the UPPAAL model checker
- Two Boolean variables: x and y, one **clock**: c
- The state machine on the left changes x nondeterministically
- The state machine on the right assigns y to x with a delay of at most 10 time units

Formal modeling with timed automata (2)



- Continuous time
- All **clocks** in the model progress synchronously
- Transitions: either a discrete transition or a timed transition (only clocks progress)
- Each clock can be reset by a transition

Linear temporal logic (LTL)

- Formal language that extends the usual propositional Boolean logic
- Variables: **atomic propositions**, e.g., p and q
- Usual Boolean operators are allowed

Linear temporal logic (LTL)

- Formal language that extends the usual propositional Boolean logic
- Variables: **atomic propositions**, e.g., p and q
- Usual Boolean operators are allowed
- **Temporal operators**, such as:
  - $\mathbf{G}(p \wedge \neg q)$ means "$p \wedge \neg q$ holds in each element of the state sequence" ("**g**lobally")

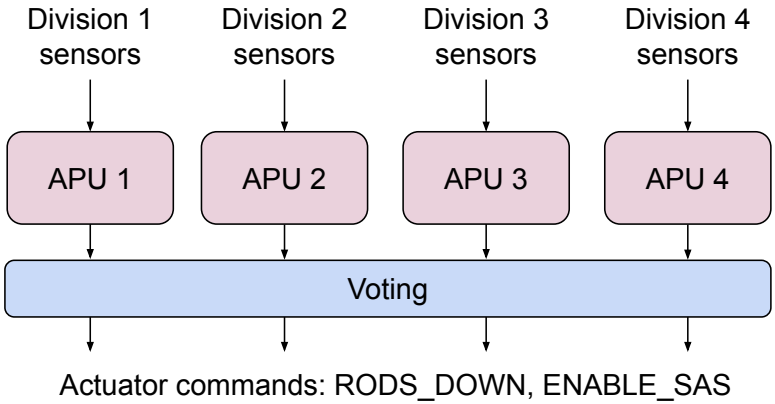
Linear temporal logic (LTL)

- Formal language that extends the usual propositional Boolean logic
- Variables: **atomic propositions**, e.g., p and q
- Usual Boolean operators are allowed
- **Temporal operators**, such as:
  - $\mathbf{G}(p \wedge \neg q)$ means "$p \wedge \neg q$ holds in each element of the state sequence" ("**g**lobally")
  - $\mathbf{F}(p \wedge \neg q)$ means "$p \wedge \neg q$ holds for at least one element of the state sequence" (in the **f**uture)

Linear temporal logic (LTL)

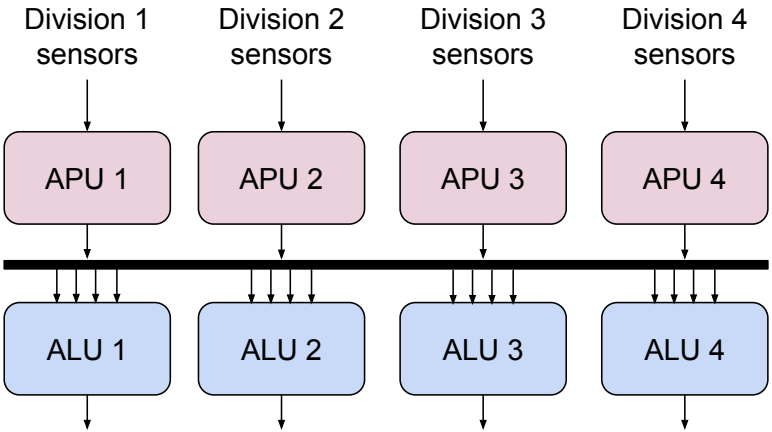
- Formal language that extends the usual propositional Boolean logic
- Variables: **atomic propositions**, e.g., p and q
- Usual Boolean operators are allowed
- **Temporal operators**, such as:
  - $\mathbf{G}(p \wedge \neg q)$ means "$p \wedge \neg q$ holds in each element of the state sequence" ("**g**lobally")
  - $\mathbf{F}(p \wedge \neg q)$ means "$p \wedge \neg q$ holds for at least one element of the state sequence" (in the **f**uture)
- In metric interval temporal logic (MITL), **G** and **F** can be annotated with time intervals

Case study: reactor protection I&C system



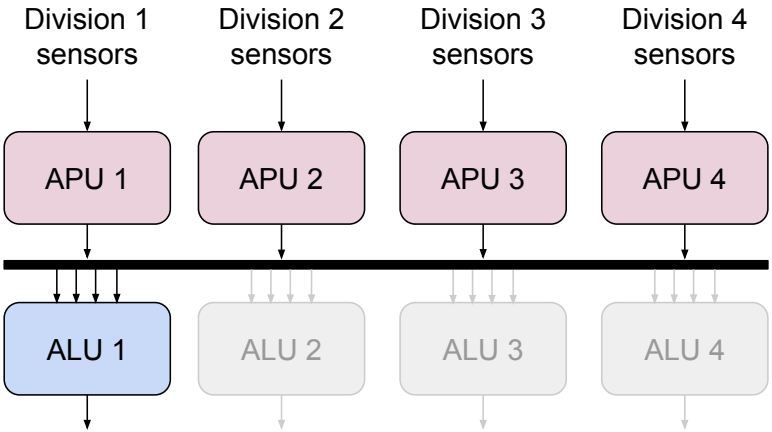
- Part of reactor shutdown safety function
- Failure tolerance implemented with redundancy
- 4 identical divisions in different buildings
- Acquisition and processing units (APUs) process sensor measurements

Reactor protection I&C system: voting



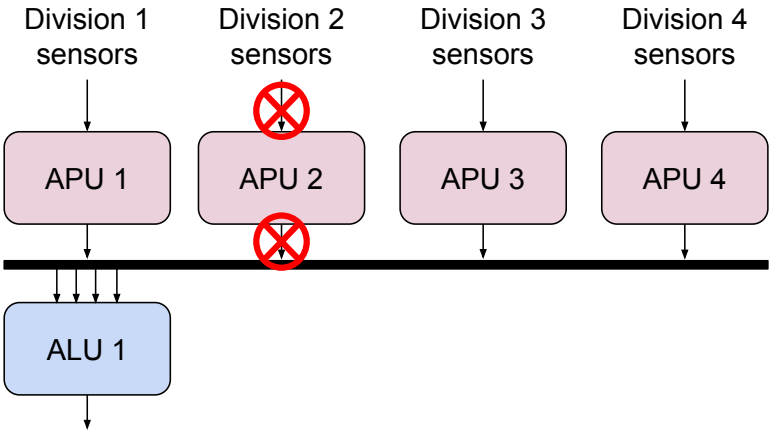
- Actuation logic units (ALUs), or voting units
- Each APU is connected to each ALU

Modeling simplification (2019)



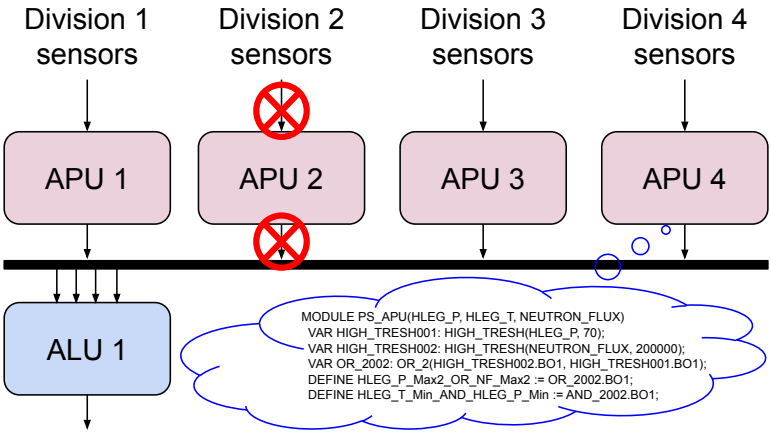
- When verifying the system, we are interested in the output of only one ALU
- Due to symmetry, verification of all ALUs is equivalent
- We can retain only one ALU

Failure modeling (2019)



- "N+1" criterion: the safety function (here, reactor shutdown) shall tolerate arbitrary failures in single division
- Due to symmetry, can choose either division 2, 3 or 4
- Here, we chose division 2

Modeling in more detail

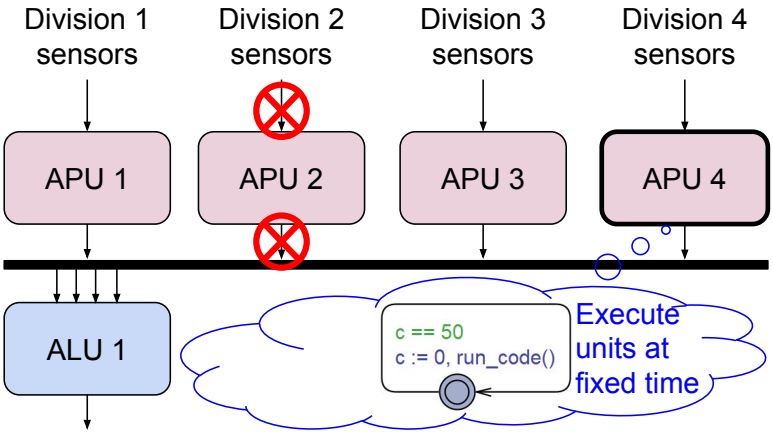


- Each unit is a network of basic blocks
- Can be modeled graphically, then represented in the **nuXmv** language

Why real-time modeling?

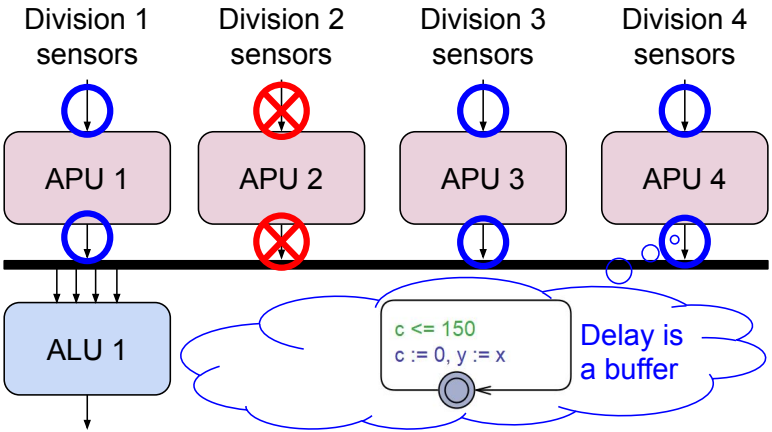
- Our previous work
  - Discrete time (number of executed transitions)
  - Synchronous execution of all units
  - Communication delays are multiples of the cycle time
- Disadvantages
  - In reality, execution of units is not synchronous
  - In reality, delays may not be multiples of the cycle time
  - Cannot model-check properties with explicitly specified time durations
- Real-time modeling solves these problems

Real-time modeling of units



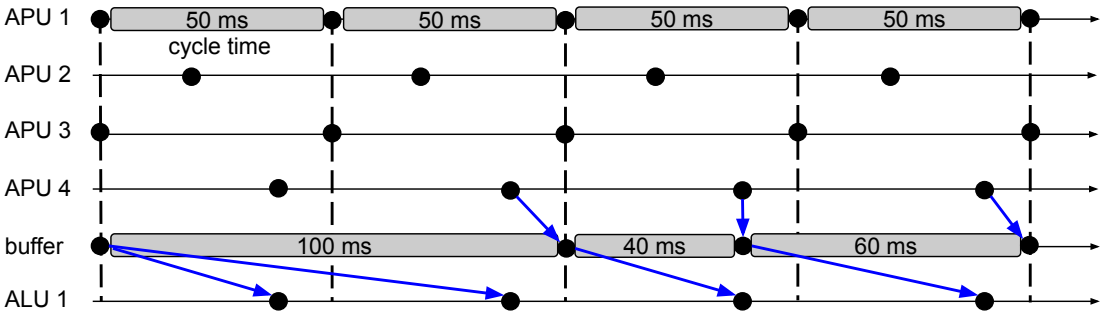
- In reality, the execution of all units is cyclic
- E.g., cycle time = 50 ms
- Execute the unit only when the time is appropriate
- Clock values of different units may be different

Real-time modeling of communication delays



- In reality, an output of one unit passes through several buffers before it reaches another unit
- We model this delay as one buffer
- The bounds on the delay can be estimated

Example of a timeline



- APU 4 $\rightarrow$ ALU 1 communication is illustrated

Example of a real-time requirement

$$(\mathbf{G}\neg\texttt{MAN\_RESET}_1) \rightarrow \mathbf{G}((\mathbf{G}_{[0,150]} \vee_{i=1}^{4} (\texttt{HLEG\_P}_i > 70)) \rightarrow \mathbf{F}_{[0,300]}\texttt{RODS\_DOWN}_1)$$

if manual reset of ALU 1 is **never** requested, **then**
always,
if the hot leg pressure (HLEG_P) is above 70 bar in any division **during the next** 150 ms,
then ALU 1 will produce the RODS_DOWN command at least once **within the next** 300 ms

- Metric interval temporal logic (MITL)
- This requirement is violated, but replacing $\vee$ with $\wedge$ makes it satisfied

Experiments: non-real-time requirements (total: 20)

Model checking algorithm		BMC	IC3
Checked properties	Total	13	11
	Satisfied	0	7
	Violated	13	4
	Median processing time (s)	0.5	89.0
	Maximum bound processed	11	12
Unchecked properties	Total	7	9
	Unchecked due to time limit	6	9
	Unchecked due to memory limit	1	0
	Minimum bound processed	32	198

Time limit: one hour per temporal property

Experiments: real-time requirements (total: 28)

Model checking algorithm		BMC	IC3
Checked properties	Total	22	0
	Satisfied	0	0
	Violated	22	0
	Median processing time (s)	2.2	—
	Maximum bound processed	15	—
Unchecked properties	Total	6	28
	Unchecked due to time limit	6	28
	Unchecked due to memory limit	0	0
	Minimum bound processed	19	15

Time limit: one hour per temporal property

Conclusions

- Approach to model and verify fault-tolerant real-time nuclear I&C systems
- Based on timed automata and the nuXmv model checker
- Can verify timed (i.e., with specific time durations specified) temporal requirements
- However, this is only possible with bounded model checking (BMC)

Future work

- Try on a larger case study
  - E.g., the one from our previous work [Buzhinsky I., Pakonen A. Model-checking detailed fault-tolerant nuclear power plant safety functions. IEEE Access, 2019]
- Integrate the approach into our model generation tool
- Examine whether verification would be feasible with precise modeling of network communication

Questions?

Thank you for your attention!

Igor Buzhinsky

igor.buzhinskii@aalto.fi

Timed model checking of fault-tolerant nuclear I&C systems