Properties of concurrent programs

- What is a concurrent program?
- Typical properties
Contents of the lecture

Finite state automata

• What is a finite state automaton?
• Examples of automata and their properties
Contents of the lecture

System model verification

- System model verification
- Most important properties to verify
- An algorithm of the system model verification
  - A simple example
Properties of concurrent programs

- What is a concurrent program?
- Typical properties
What is a concurrent program?

- A concurrent program consists of processes, where:
  - the processes run at the same time;
  - the processes may share some resources, e.g. variables;
  - the processes may interact with each other.
Properties of concurrent programs

Typical properties

- A concurrent program P consists of concurrently running processes: P1, P2, P3, ...
- Properties of P are: $q_1$, $q_2$, $q_3$, $q_4$, ...
- A property $q_i$ of the program P may be specified by a formula $\phi_i$, which will be satisfied in every execution of P (from the first state, in which $s_0 \models \phi_i$), independently of the way of managing the processes.
Properties of concurrent programs

Typical properties

- **safety** — $q_1$ is satisfied **at every moment**
  \[ \phi_1 \equiv Gq_1 \]  
e.g. access to a critical section CS is always possible.

- **reachability** — $q_2$ will be satisfied **at some moment**
  \[ \phi_2 \equiv Fq_2 \]  
e.g. a process will eventually get to the CS.

- **response** — $q_3$ is satisfied **from time to time**
  \[ \phi_3 \equiv GFq_3 \]  
e.g. a process is set to sleep from time to time.

- **persistence** — **from some moment:**
  \[ q_4 \text{ is satisfied at every moment} \]
  \[ \phi_4 \equiv FGq_4 \]  
e.g. from some moment: a process wants no access to the CS.

- **liveness** — $q_5$ is reachable **as a result** of $\rho$
  \[ \phi_5 \equiv G(\rho \Rightarrow Fq_5) \]  
e.g. a process accesses the CS on demand.
Finite state automata

- What is a finite state automaton?
- Examples of automata and their properties
What is a finite state automaton?

- is an abstract state machine,
- consists of a finite number of states and transitions between them,
- has initial states and may have final states,
- is an automaton, in which transitions between states are fired deterministically (interchangeably described by a transition function).
What is a finite state automaton?

Automaton $A=\{\Sigma, S, S_0, \rho, F\}$ is defined by:

- $\Sigma$ – an alphabet (a set of states of the program),
- $S$ – a set of states of the automaton,
- $S_0$ – a set of initial states,
- $\rho$ – a transition function ($S \times \Sigma \times S$),
- $F$ – a set of final states.

A trace is a path of states, e.g. $s_1, s_2, s_2, s_3$.

A state may be described by propositions, that are true in it.
Examples of automata and their properties

An automaton closing the door

- Goal: to automatically close the door, when it is opened.
- Assumptions:
  - at the beginning the door is opened,
  - the door closes instantly (the closing time equals zero).
- Proposition $p$ — the door is closed.
- At the moment ($s_1$) of turning the automaton on it is true, that:
  $\neg p \land XGp$
Finite state automata

Examples of automata and their properties

An automaton setting the light on and off

- Goal: to automatically set the light on at 18:00 and off at 6:00.
- Assumptions:
  - at the beginning the light is off,
  - the light is set on at 18:00 and off at 6:00,
  - setting the light on and off is instant,
  - the clock H starts at 12:00.
- Proposition $p$ — the light is on.
Finite state automata

Examples of automata and their properties

An automaton setting the light on and off

- At the moment ($s_1$) of turning the automaton on it is true, that:

$$\neg p \land H=12 \land G(\neg p \land H\neq 18 \Rightarrow X\neg p) \land G(\neg p \land H=18 \Rightarrow Xp) \land G(p \land H\neq 6 \Rightarrow Xp) \land G(p \land H=6 \Rightarrow X\neg p)$$

- The automaton satisfies the following properties:
  - the lamp works: $GFp \land GF\neg p$
  - the lamp does not twinkle: $\neg F(p \land X\neg p \land XXp)$
Examples of automata and their properties

Automata of concurrent processes

- A process $P_1$ places lines of text in a table “lines”.
- A process $P_2$ removes lines from “lines”, if they are incorrect.
- $P_1$ and $P_2$ run concurrently.

$P_1$: \(\text{while } (\text{true}) \text{ do } \{ \text{lines.add(new line); } \} \)

$P_2$: \(\text{while } (\text{true}) \text{ do } \{ \text{if(lines.last.incorrect) } \{\text{lines.last.remove(); } \} \} \)
Examples of automata and their properties

Automata of concurrent processes

Are these properties satisfied?

• “lines” never contains more than one incorrect line:
  \[ \neg F(\text{lines.incorrect.count} > 1) \]

• every line exists in “lines” for some time:
  \[ (\forall \text{line})(F(\text{lines.contains(line)})) \]
Finite state automata

Examples of automata and their properties

Automata of concurrent processes

Are these properties satisfied?

- “lines” never contains more than one incorrect line:
  \[ \neg F(\text{lines.incorrect.count} > 1) \]  NO

- every line exists in “lines” for some time:
  \[ (\forall \text{ line})(F(\text{lines.contains(line)})) \]  YES
System model verification

- System model verification
- Most important properties to verify
- An algorithm of the system model verification
  - A simple example
System model verification

Input data:
- a formal model of system functions given as an automaton;
- properties, that must be satisfied by the system, given as temporal logic formulas.

Output data:
- The answer, whether the system satisfies these properties.

*Does every possible sequence of states of the system (program) satisfy the properties?*

*Does any possible sequence of states of the system (program) satisfy the properties?*
System model verification

Most important properties to verify

• reachability
  LTL: $Fp$
  CTL: $EFp$
  – the “wanted” state $p$ of the system will eventually be reached,

• safety
  LTL: $G\neg q$
  CTL: $AG\neg q$
  – the “unwanted” state $q$ of the system will never be reached.

Goal: to verify (automatically) whether a given formula is true in a given model.
System model verification

An algorithm of the system model verification

1) Build a finite state automaton $A_S$ for the system's model $S$.
2) Write the system's properties as a formula $f$.
3) Build a finite state automaton $A_{\neg f}$ for the formula $\neg f$.
4) Build a product automaton for $A_P = A_S \times A_{\neg f}$.
5) Verify (automatically) whether $A_P$ exists.
System model verification

An algorithm of the system model verification

- $A_{\neg f}$ should accept only such sequences of states of the program of the system $S$, that satisfy the formula $\neg f$.
- If there exists such a sequence of states in $A_S$, that corresponds to the formula $\neg f$, then such a run of the program is possible, that does not satisfy the given properties.
A simple example

- An automaton $A_S$ models a system, where every state is labelled by all these propositions, that are true in it: $a$ and $b$. ($\emptyset$ – $a$ and $b$ are false)

- Is the property $f \equiv EFb$ satisfied for $A_S$?

- The automaton $A_{\neg f}$ models the formula $\neg f \equiv \neg EFb \equiv AG\neg b$.

- In $A_S$ there does not exist any infinite trace $A_{\neg f}$ – the property $f$ is satisfied.

Notice, that the property $AFb$ is satisfied too.
The end

**Literature:**