Information Systems Analysis

Temporal Logic and Timed Automata

(4)

System model verification

© Paweł Głuchowski, *Wrocław University of Technology*

version 2.1

Contents of the lecture

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

Properties of concurrent programs

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 ϕ_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 <u>at every moment</u>

  $\phi_1 \equiv Gq_1$ e.g. access to a critical section CS is always possible.

- **reachability** – q_2 will be satisfied <u>at some moment</u>

  $\phi_2 \equiv Fq_2$ e.g. a process will eventually get to the CS.

- **response** – q_3 is satisfied <u>from time to time</u>

  $\phi_3 \equiv GFq_3$ e.g. a process is set to sleep from time to time.

- **persistence** – <u>from some moment</u>:
  q_4 is satisfied <u>at every moment</u>

  $\phi_4 \equiv FGq_4$ e.g. from some moment: a process wants no access to the CS.

- **liveness** – q_5 is reachable <u>as a result</u> of p

  $\phi_5 \equiv G(p \Rightarrow Fq_5)$ e.g. a process accesses the CS on demand.

Finite state automata

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

Finite state automata

What is a finite state automaton?

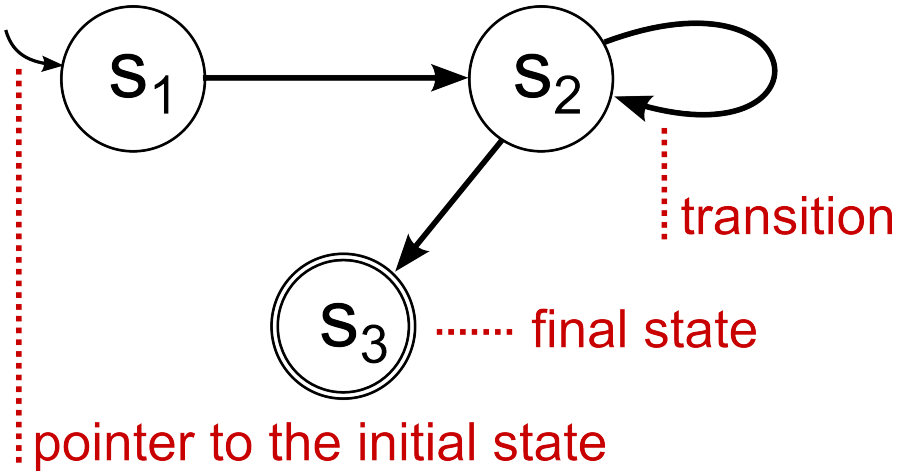
- is an abstract state machine,
- consists of a finite number of states and transitions between them,
- has initial states and may have final states,
- is an automaton, in which transitions between states are fired deterministically
  (interchangeably described by a transition function).

Finite state automata

What is a finite state automaton?

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

- Σ – an alphabet (a set of states of the program),
- S – a set of states of the automaton,
- S_0 – a set of initial states,
- ρ – 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.

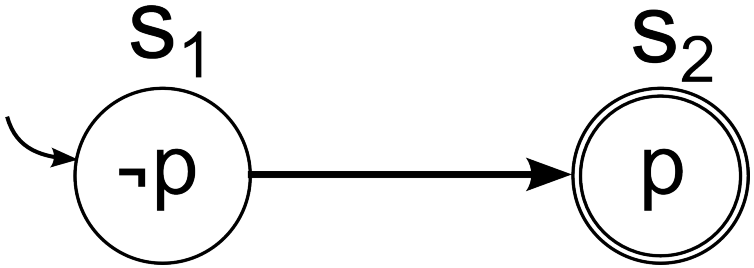
A state may be described by propositions, that are true in it.

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:

  $\neg p \wedge XGp$

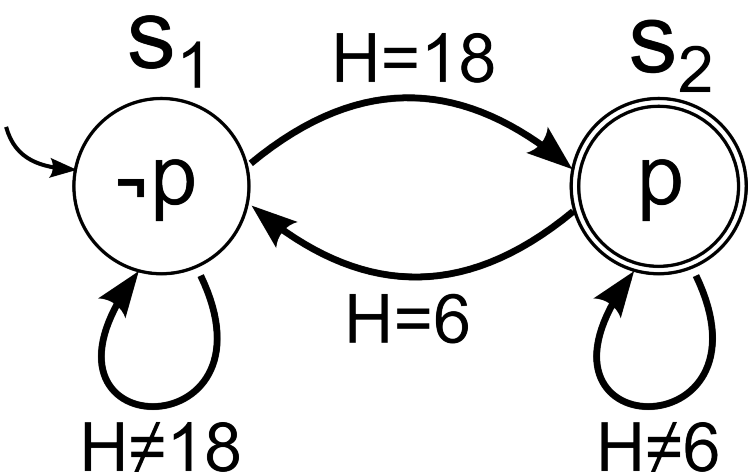


Finite state automata

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

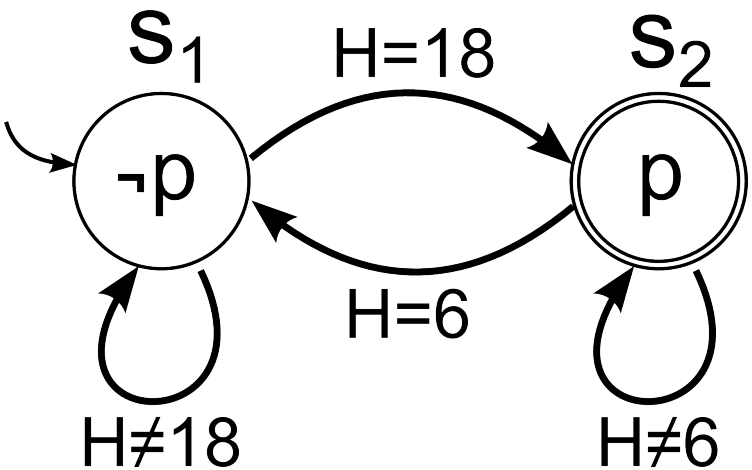
Examples of automata and their properties

An automaton setting the light on and off

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

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

- The automaton satisfies the following properties:
  - the lamp works:
    $GFp \wedge GF\neg p$
  - the lamp does not twinkle:
    $\neg F(p \wedge X\neg p \wedge XXp)$

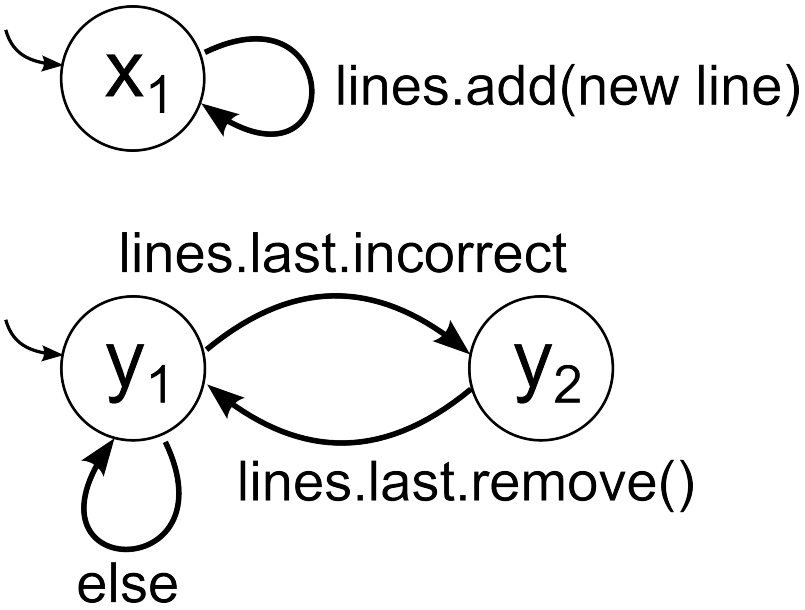


Finite state automata

Examples of automata and their properties

Automata of concurrent processes

- A process **P1** places lines of text in a table "*lines*".
- A process **P2** removes lines from "*lines*", if they are incorrect.
- P1 and P2 run concurrently.



P1: *while (true) do { lines.add(new line); }*

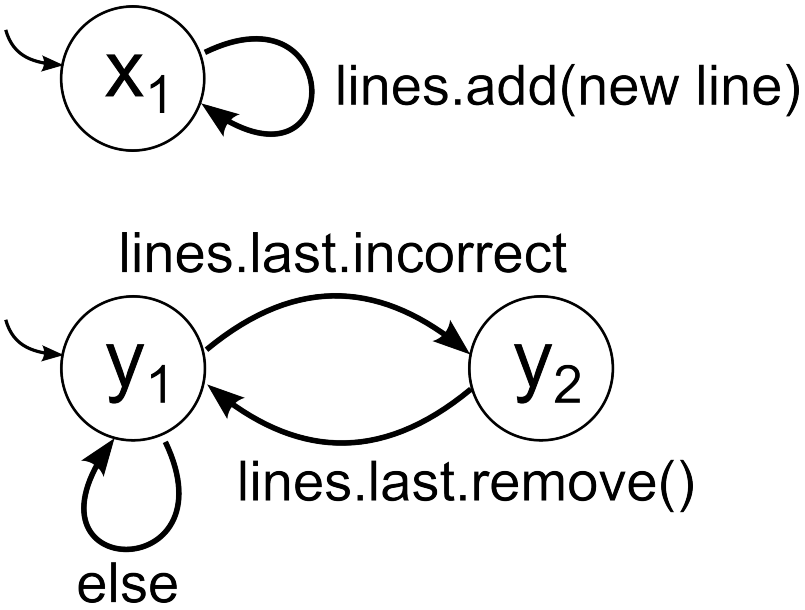
P2: *while (true) do { if(lines.last.incorrect) {lines.last.remove(); }}*

Finite state automata

Examples of automata and their properties

Automata of concurrent processes

Are these properties satisfied?



- *"lines"* never contains more then one incorrect line:

  ¬F(lines.incorrect.count > 1)

- every line exists in "lines" for some time:

  (∀ line)(F(lines.contains(line)))

Finite state automata

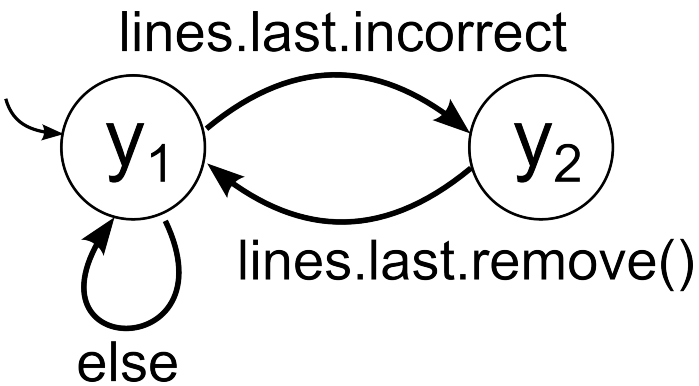
Examples of automata and their properties

Automata of concurrent processes

Are these properties satisfied?

- *"lines"* never contains more then one incorrect line:

  ¬F(lines.incorrect.count > 1) NO



- every line exists in "lines" for some time:

  (∀ line)(F(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: $\mathsf{F}p$
  CTL: $\mathsf{EF}p$
  – the "wanted" state *p* of the system will <u>eventually</u> be reached,

- **safety**
  LTL: $\mathsf{G}\neg q$
  CTL: $\mathsf{AG}\neg q$
  – the "unwanted" state *q* of the system will <u>never</u> 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

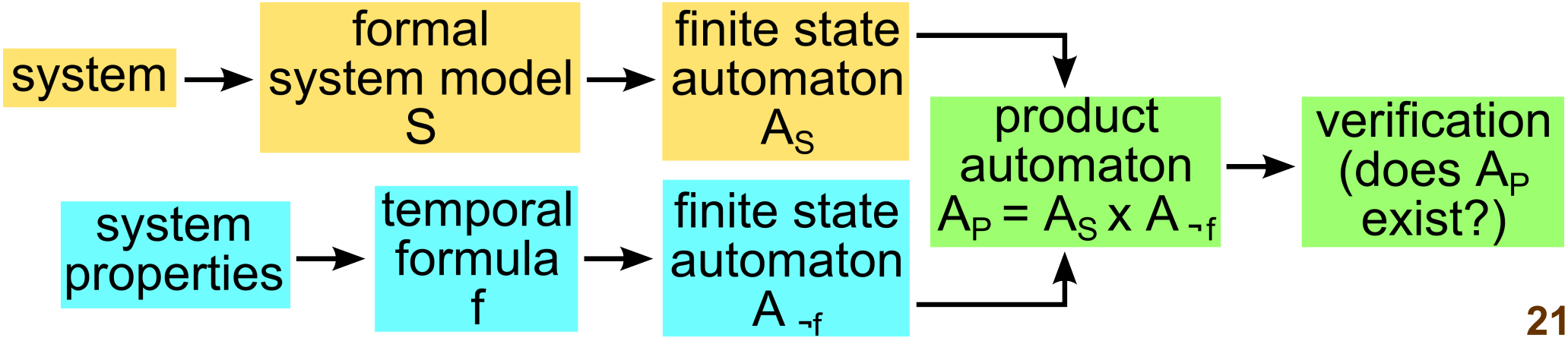
1) Build a finite state automaton A_S for the system's model S.

2) Write the system's properties as a formula f.

3) Build a finite state automaton $A_{\neg f}$ for the formula ¬f.

4) Build a product automaton for $A_P = A_S \times A_{\neg f}$.

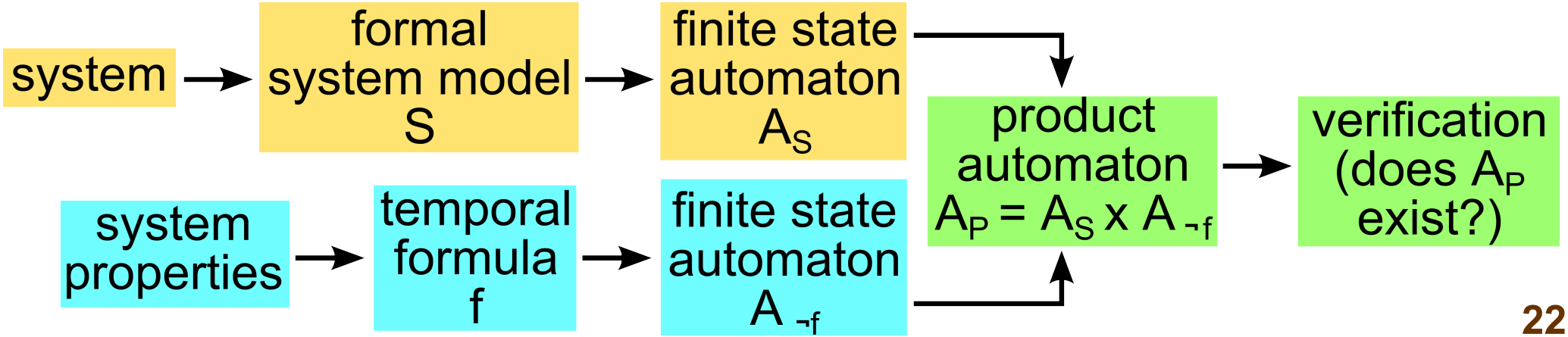
5) Verify (automatically) whether A_P exists.



System model verification

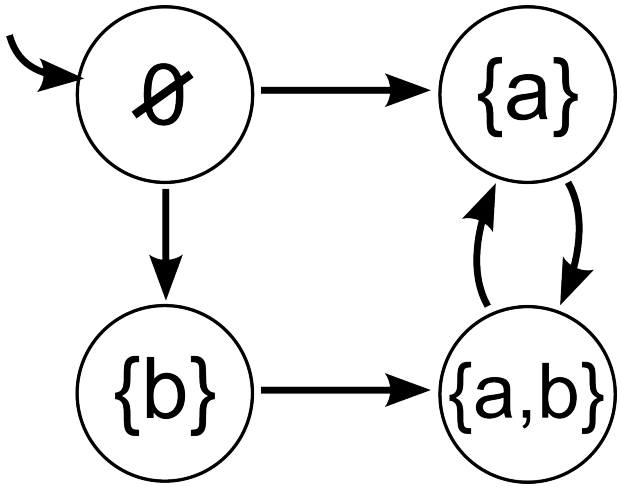
An algorithm of the system model verification

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

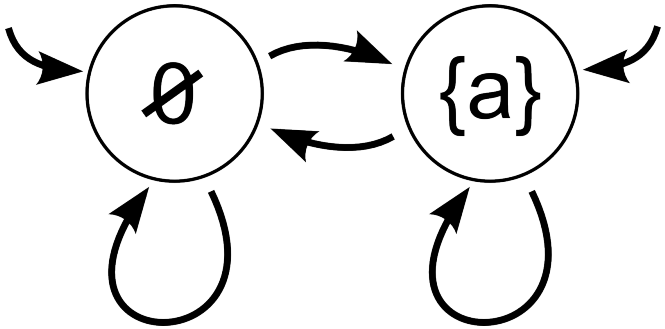


System model verification

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*. (∅ – *a* and *b* are false)
- Is the property $f \equiv EF\mathit{b}$ satisfied for A_S?
- The automaton $A_{\neg f}$ models the formula $\neg f \equiv \neg EF\mathit{b} \equiv AG\neg\mathit{b}$.
- In A_S there does not exist any infinite trace $A_{\neg f}$ – the property f is satisfied.



Notice, that the property AF*b is satisfied too.*

The end

Literature:

- R. Alur et al. “Automata for modelling real-time systems”, 1990
- J.E. Hopcroft, J.D. Ullman “Introduction of Automata Theory, Languages, and Computation”, 2001
- R. Klimek “*Zastosowanie logiki temporalnej w specyfikacji i weryfikacji oprogramowania – w stronę czasu rzeczywistego”*, in: “Wykłady zaproszone. Systemy Czasu Rzeczywistego 2000”, 2001