# Concurrency
week 6

Anders Møller
amoeller@daimi.au.dk

## Agenda

- Model checking **regular safety properties** with *finite-state automata*
- Model checking **ω-regular properties** with *Büchi automata*



## Model checking

$T \models P$

the model (a transition system) — satisfies — the property (a linear-time property)

Goal: obtain decision procedures for interesting classes of linear-time properties

- **regular** (safety) properties
- **ω-regular** properties — this week
- **LTL**-expressible properties — next week



## The LTSA approach to safety checking
**(week 2)**

- A **safety property** *P* defines a deterministic process that asserts that any trace including actions in the alphabet of *P*, is accepted by *P*
- **Algorithm sketch**:
  1. Compile the FSP model composed with the safety property into a labeled transition system (LTS)
  2. Check whether the ERROR state is reachable from the initial state
  3. If reachable, a path corresponds to a counterexample execution trace; otherwise, the model satisfies the property

***Let's explain this in more detail, using the terminology of [B&K] and regular languages!***

## Regular safety properties

- Let $P$ be a safety property over $AP$
- Every trace $\sigma$ that violates $P$ has a finite *bad prefix* $\rho \in pref(\sigma)$ where $\forall \theta \in (2^{AP})^{\omega}: \rho\theta \notin P$
- Let $BadPref(P) = \{ \rho \in pref(\sigma) \mid \sigma \in (2^{AP})^{\omega} \setminus P \ \wedge \ \forall \theta \in (2^{AP})^{\omega}: \rho\theta \notin P \}$

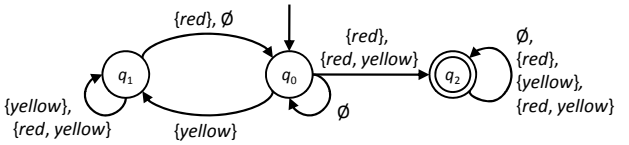
- $P$ is a ***regular safety property*** if $BadPref(P)$ is regular

- The class of regular safety properties corresponds to the class of *finite-state automata* (over the alphabet $2^{AP}$)



## Example: traffic light

- The following NFA (nondeterministic finite automaton) accepts the bad prefixes of the safety property "a red phase must be preceded immediately by a yellow phase"
- Let $\Sigma = 2^{AP}$, $AP = \{yellow, red\}$




- Note that the atomic propositions are here on the *transitions*, not on the states!



## Checking safety properties

- Let $T$ be a TS and $P$ be a safety property
- Define $Traces_{fin}(T) = pref(Traces(T))$
  (i.e. $Traces_{fin}(T)$ is the set of finite prefixes of traces of $T$)

- Lemma:
  $T \models P \ \Leftrightarrow \ Traces_{fin}(T) \cap BadPref(P) = \emptyset$
- *Proof?* (see [B&K] p.114)



## Checking *regular* safety properties

- Let $T$ be a TS and $P$ be a *regular* safety property
- Let $A$ be an NFA with $\mathcal{L}(A) = BadPref(P)$

- $T \models P \ \Leftrightarrow \ Traces_{fin}(T) \cap \mathcal{L}(A) = \emptyset$

- To decide this, we first introduce the notion of *invariants…*

## Invariant properties

- An ***invariant*** is a linear-time property that can be expressed on the form
  $\{\sigma_0\sigma_1\sigma_2\ldots \in (2^{AP})^\omega \mid \forall i\colon \sigma_i \models \varphi\}$
  for some propositional logic formula $\varphi$
  (i.e. $\varphi$ contains no temporal operators)
- Example:
  *MUTEX* = $\{\rho \in (2^{AP})^\omega \mid \rho$ does not contain {crit1, crit2}$\}$
  is the invariant defined by $\varphi = \neg$crit1 $\wedge \neg$crit2
- Every invariant is also a safety property *(Proof?)*



## Checking invariants and finding counterexamples

- Let $T$ be a finite TS and $P$ be an invariant specified by a propositional logic formula $\varphi$
- $T \models P \Leftrightarrow \mathit{Traces}_{fin}(T) \cap \mathit{BadPref}(P) = \emptyset$
  $\Leftrightarrow \forall \sigma_0\sigma_1\sigma_2\ldots\sigma_n \in \mathit{Traces}_{fin}(T)\colon \sigma_n \models \varphi$
- If $\exists \sigma_0\sigma_1\sigma_2\ldots\sigma_n \in \mathit{Traces}_{fin}(T)\colon \sigma_n \not\models \varphi$
  then $\forall \theta \in (2^{AP})^\omega\colon \sigma_0\sigma_1\sigma_2\ldots\sigma_n\theta \not\models P$
  (i.e. $\sigma_0\sigma_1\sigma_2\ldots\sigma_n\theta$ is a counterexample for any $\theta$)
- Algorithm: breadth-first search through $T$ for a state where $\varphi$ is violated… (see [B&K] Sec.3.3.1)



## A reduction from regular safety checking to invariant checking

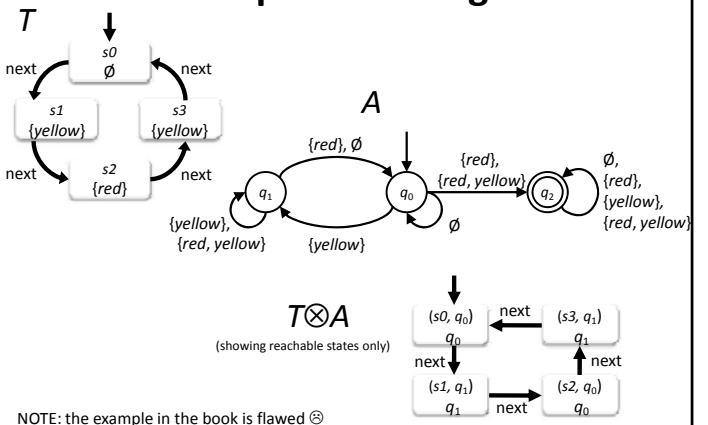
- Given a TS $T$ and an NFA $A$ with alphabet $2^{AP}$ we want to construct a TS $T \otimes A$ such that

$$\mathit{Traces}_{fin}(T) \cap \mathcal{L}(A) = \emptyset \quad \Updownarrow \quad T \otimes A \models P_{inv(A)}$$

  where $P_{inv(A)}$ is an invariant that depends on $A$
- (We'll define $P_{inv(A)}$ soon…)



## Example: traffic light




NOTE: the example in the book is flawed ☹

## Product of TS and NFA

- Let $T = (S, Act, \Rightarrow, I, AP, L)$ be a TS (without terminal states, as usual) and $A = (Q, \Sigma, \delta, Q_0, F)$ be an NFA (in this version, having a *set* of initial states) where $\Sigma = 2^{AP}$ and $Q_0 \cap F = \emptyset$
- The ***product transition system*** $T \otimes A$ is the TS $(S', Act, \Rightarrow', I', AP', L')$ where
  - $S' = S \times Q$
  - $\Rightarrow'$ is defined by: $\dfrac{(s, \alpha, t) \in \Rightarrow \qquad p \in \delta(q, L(t))}{((s, q), \alpha, (t, p)) \in \Rightarrow'}$
  - $I' = \{ (s_0, q) \mid s_0 \in I \wedge \exists q_0 \in Q_0 : q \in \delta(q_0, L(s_0)) \}$
  - $AP' = Q$
  - $L'(s, q) = \{q\}$ for all $s \in S$ and $q \in Q$

13

## The invariant $P_{inv(A)}$

- Choose $P_{inv(A)}$ as the invariant defined by

  $\bigwedge_{q \in F} \neg q$ (where $F$ is the accept states of $A$)

- This invariant is satisfied if none of $A$'s accept states are reachable in $T \otimes A$

14

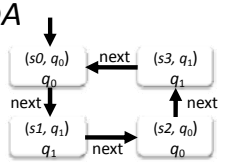
## Putting the pieces together…

- Let $T$ be a TS over $AP$, let $P$ be a regular safety property over $AP$, and let $A$ be an NFA where $\mathcal{L}(A) = BadPref(P)$
- Theorem:
  The following statements are equivalent:
  *a)* $T \models P$
  *b)* $Traces_{fin}(T) \cap \mathcal{L}(A) = \emptyset$
  *c)* $T \otimes A \models P_{inv(A)}$
- *Proof?* (see [B&K] p.167)

15

## Example: traffic light

$T \otimes A$




- $T \otimes A$ has no reachable state whose label is an accept state in $A$
- so we conclude that $T \models P$ ☺

16

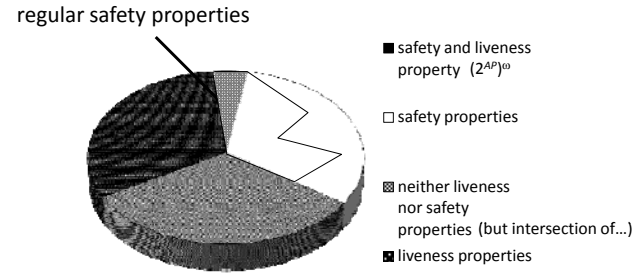
## Summary of model checking for regular safety properties

Let *T* be a **finite TS** and let *P* be a **regular safety property** described by an NFA *A* (i.e. $\mathcal{L}(A) = BadPref(P)$)

1. Construct $T \otimes A$
2. If $T \otimes A$ has a reachable state (*s*, *q*) where *q* is an accept state in *A*, then report "**NOT SATISFIED!**", find a (shortest possible) path from the initial state to such a state, and report the corresponding trace fragment as a counterexample
3. Otherwise, report "**SATISFIED!**"

17

## Classification of LT properties




- Not all safety properties are regular
- (Study this in exercises….)

18

## Agenda

- Model checking **regular safety properties** with *finite-state automata*
- Model checking **ω-regular properties** with *Büchi automata*

19

## The LTSA approach to progress checking with 'fair choice' (week 2)

1. Compile the FSP program into a labeled transition system (LTS)
1. Search for terminal sets (*A set of states S is a **terminal set** if every state in S is reachable from every other state in S via one or more transitions, and there is no transition from within S to any state outside S*)
2. Check for each terminal set that at least one of the progress set actions occurs as a transition (if not, report a path to the terminal set and its actions)

***Let's generalize this to encompass "regular" liveness properties, using the [B&K] terminology***

20

## Model checking liveness properties

- An LT property $P$ is a ***liveness property*** if $pref(P) = (2^{AP})^*$
- Liveness is fundamentally about *infinite* traces, so ordinary automata (that accept sets of *finite* strings) are useless here
- Let's introduce a kind of automata that work on sets of *infinite* strings!

21

## Büchi automata

A **nondeterministic Büchi automaton** (NBA) is a 5-tuple $(Q, \Sigma, \delta, Q_0, F)$ where

- Q is a finite set of ***states***
- $\Sigma$ is an ***alphabet***
- $\delta: Q \times \Sigma \rightarrow 2^Q$ is a ***transition function***
- $Q_0 \subseteq Q$is a set of ***initial states***
- $F \subseteq Q$ is a set of ***accept states***

– so far, it looks just like an NFA!

22

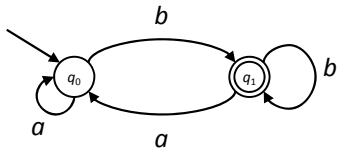
## The language of a Büchi automaton

- Let $A = (Q, \Sigma, \delta, Q_0, F)$ be an NBA and $\sigma = a_0a_1a_2\ldots \in \Sigma^\omega$
- A ***run*** for $\sigma$ is an infinite sequence of states $q_0q_1q_2\ldots$ where
  - $q_0 \in Q_0$ and
  - $q_{i+1} \in \delta(q_i, a_i)$ for all $i$
- Such a run is ***accepted*** by $A$ if $q_i \in F$ for infinitely many $i$
- The ***language*** of $A$, denoted $\mathcal{L}(A)$, is the set of infinite strings $\sigma \in \Sigma^\omega$ where some run is accepted by $A$

23

## Example

What is the language of this Büchi automaton?




24

## Nonblocking NBA

- An NBA $(Q, \Sigma, \delta, Q_0, F)$ is is ***nonblocking*** if $\delta(q, a) \neq \emptyset$ for all $q \in Q$ and $a \in \Sigma$
- For a nonblocking NBA, every infinite string over the same alphabet has at least one run
- We can assume without loss of generality that our NBAs are nonblocking (Why?)

25

## ω-regular expressions

- As a variant of Kleene's theorem, Büchi automata correspond to "ω-regular expressions"!
- An ***ω-regular expression*** $G$ over $\Sigma$ has the form
  $G = E_1 F_1^{\omega} + \ldots + E_n F_n^{\omega}$
  where $E_1, \ldots, E_n, F_1, \ldots, F_n$ are (ordinary) regular expressions over $\Sigma$ and $\Lambda \notin \mathcal{L}(F_i)$ for all $i$
- The ***language*** of $G$ is
  $\mathcal{L}(G) = \mathcal{L}(E_1) \cdot \mathcal{L}(F_1)^{\omega} \cup \ldots \cup \mathcal{L}(E_n) \cdot \mathcal{L}(F_n)^{\omega}$
- A language $L \subseteq \Sigma^{\omega}$ is ***ω-regular*** if it is the language of some ω-regular expression

26

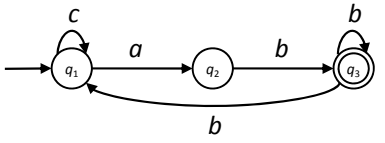
## Properties of Büchi automata and ω-regular languages

- $L \subseteq \Sigma^{\omega}$ is ω-regular iff it is the language of some NBA (Theorem 4.32 in [B&K], exercises…)
- The class of ω-regular languages is closed under union (trivial), intersection (next week), and complement (difficult!)
- Deterministic Büchi automata are strictly less expressive than nondeterministic Büchi automata!
- Minimizing Büchi automata is PSPACE-hard (but heuristics exist)

27

## Example

What is an ω-regular expression that has the same language as this Büchi automaton?




Hint: it must have the form $E_1 F_1^{\omega} + \ldots + E_n F_n^{\omega}$

$c^*ab(b^+ + bc^*ab)^{\omega}$

28

## ω -regular properties

- Let *P* be a linear-time property over *AP*

- *P* is an ***ω-regular property*** if it is an ω-regular language

- (Compare with the definition of *regular safety property*)

29

## Regular vs. ω-regular?

- Any regular safety property *P* is also an ω-regular property!
- *Proof?*

  The complement of *P*,
  $(2^{AP})^{\omega} \setminus P = BadPref(P) \cdot (2^{AP})^{\omega}$
  is ω-regular, and the class of ω-regular languages is closed under complement

30

## Examples (from last week)

- Peterson:
  - "Each process will eventually enter its critical region"
  - "Each process will enter its critical region infinitely often"
  - "Each waiting process will eventually enter its critical region"
- Vending machine:
  - "The machine will always eventually serve a drink"
  - "The machine will always eventually serve coffee"
- Are these liveness properties all ***ω-regular***?

31

## Checking ω-regular properties

- Let *T* be a TS and *P* be an ω-regular property
- $T \models P \Leftrightarrow Traces(T) \cap (2^{AP})^{\omega} \setminus P = \emptyset$

- For regular safety properties, we introduced *invariants* and *reduced* regular safety checking to invariant checking
- Now, we introduce *persistence properties* and reduce ω-regular checking to persistence checking!

32

## Persistence properties

- An ***persistence property*** is a linear-time property that can be expressed on the form
  $\{\sigma_0\sigma_1\sigma_2\ldots \in (2^{AP})^\omega \mid \exists i\colon \forall j \geq i\colon \sigma_j \models \varphi\}$
  for some propositional logic formula $\varphi$
- i.e. "$\varphi$ is an invariant *after a while*"

33

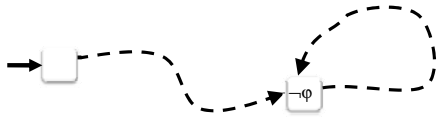
## Checking persistence properties and finding counterexamples

- Let $T = (S, Act, \Rightarrow, I, AP, L)$ be a finite TS and $P$ be a persistence property specified by a propositional logic formula $\varphi$
- Theorem: $T \not\models P \Leftrightarrow$
  $\exists s_0 s_1 s_2 \ldots s_k \ldots s_n \in S^*\colon s_0 \in I \wedge \forall 0 \leq i < n\colon \exists a \in Act\colon (s_i, a, s_{i+1}) \in \Rightarrow$
  $\wedge\ s_k = s_n \wedge L(s_k) \not\models \varphi$
  (i.e. $T$ contains a reachable state $s_k$ that belongs to a cycle and where $\varphi$ is violated)
- *Proof?* (see [B&K] p.206)
- $L(s_0)L(s_1)L(s_2)\ldots L(s_k)\ldots L(s_n)$ represents a *counterexample*

34

## Checking persistence properties and finding counterexamples

- Algorithm: *nested depth-first search* through $T$ for a state that belongs to a cycle and where $\varphi$ is violated
  (we'll skip the details!
  – if interested, see [M&K] pp.207–212)




35

## A reduction from $\omega$-regular checking to persistence checking

- Given a TS $T$ and an NBA $A$ with alphabet $2^{AP}$ we want to construct a TS $T \otimes A$ such that

$$\begin{array}{c} Traces(T) \cap \mathcal{L}(A) = \emptyset \\ \Updownarrow \\ T \otimes A \models P_{pers(A)} \end{array}$$

  where $P_{pers(A)}$ is a persistence property that depends on $A$
- (We'll define $P_{pers(A)}$ soon…)

36

## Product of TS and NBA

- Let $T = (S, Act, \Rightarrow, I, AP, L)$ be a TS and $A = (Q, \Sigma, \delta, Q_0, F)$ be a nonblocking NBA where $\Sigma = 2^{AP}$
- The ***product transition system*** $T \otimes A$ is defined exactly as for an NFA!

37

## The persistence property $P_{pers(A)}$

- Choose $P_{pers(A)}$ as the persistence property defined by

  $\bigwedge_{q \in F} \neg q$ (where $F$ is the accept states of $A$)

  (the same formula as for $P_{inv(A)}$)
- This property is satisfied if $A$'s accept states are visited only finitely many times in any execution of $T \otimes A$

38

## Putting the pieces together…

- Let $T$ be a TS over $AP$, let $P$ be an $\omega$-regular property over $AP$, and let $A$ be a nonblocking NBA where $\mathcal{L}(A) = (2^{AP})^\omega \setminus P$
- Theorem:
  The following statements are equivalent:
  *a)* $T \models P$
  *b)* $Traces(T) \cap \mathcal{L}(A) = \emptyset$
  *c)* $T \otimes A \models P_{pers(A)}$
- *Proof?* (see [B&K] p.201)

39

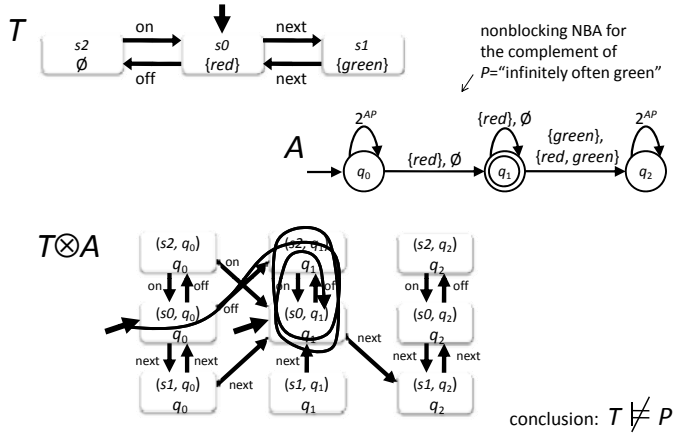
## Summary of model checking for $\omega$-regular properties

Let $T$ be a **finite TS** and let $P$ be a **$\omega$-regular property** described by a nonblocking NBA $A$ (i.e. $\mathcal{L}(A) = (2^{AP})^\omega \setminus P$ )

1. Construct $T \otimes A$
2. If $T \otimes A$ has a reachable state $(s, q)$ where $q$ is an accept state in $A$ and $(s, q)$ is on a cycle, then report "**NOT SATISFIED!**", find a path from the initial state to such a state and the cycle back to the state, and report the corresponding trace fragments as a counterexample
3. Otherwise, report "**SATISFIED!**"

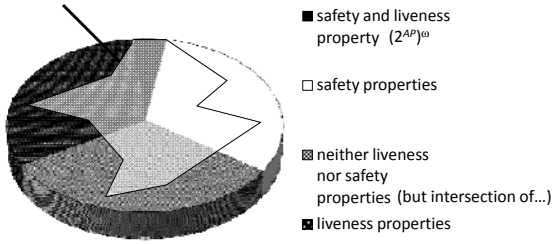
40

# Example: another traffic light




41

# Classification of LT properties




- Compare with earlier slides about "Classification of LT properties"!
- (Exercise: compare linear-time / safety / liveness / invariants / progress / persistence / LTL / regular / ω -regular properties)

42

# Summary

- Model checking with **regular safety properties** can be done with ordinary finite automata by a reduction to *invariant* model checking
  - product of the TS and an NFA representing the bad prefixes of the property
  - e.g. breadth-first search
- Model checking with **ω-regular properties** can be done with Büchi automata by a reduction to *persistence property* model checking
  - product of the TS and an NBA representing the complement of the property
  - e.g. nested depth-first search

43

## Concurrency – week 6

- Model checking **regular safety properties** with *finite-state automata*
- Model checking **ω-regular properties** with *Büchi automata*

Let $P$ be a linear-time property over $AP$

- $P$ is a ***safety property*** if $\forall \sigma \in (2^{AP})^\omega \setminus P$: $\exists \rho \in pref(\sigma)$: $\forall \theta \in (2^{AP})^\omega$: $\rho\theta \notin P$
- $P$ is a ***liveness property*** if $pref(P) = (2^{AP})^*$

- $P$ is a ***regular safety property*** if $BadPref(P)$ is regular
- $P$ is an ***invariant*** if it can be expressed on the form $\{\sigma_0\sigma_1\sigma_2\ldots \in (2^{AP})^\omega \mid \forall i: \sigma_i \models \varphi\}$ for some propositional logic formula $\varphi$
- $P$ is an ***ω-regular property*** if it is an ω-regular language
- P is a ***persistence property*** if it can be expressed on the form $\{\sigma_0\sigma_1\sigma_2\ldots \in (2^{AP})^\omega \mid \exists i: \forall j \geq i: \sigma_j \models \varphi\}$ for some propositional logic formula $\varphi$