

# Introduction to Model Checking

René Thiemann

Institute of Computer Science  
University of Innsbruck

WS 2011/2012



## Outline

- Notations
- Transition systems
- Model Checking of Linear Time Properties
- Regular Languages
  - Finite Automata
  - Büchi Automata
  - Generalized Büchi Automata

## Outline

- Notations
- Transition systems
- Model Checking of Linear Time Properties
- Regular Languages
  - Finite Automata
  - Büchi Automata
  - Generalized Büchi Automata

- atomic propositions:  $AP = \{a, b, \dots\}$
- signature:  $\Sigma = \{A, B, \dots\}$ , often  $\Sigma = 2^{AP}$
- infinite words:  $\Sigma^\omega = \{v, w, \dots\}$  where  $w = A_1 A_2 A_3 \dots$
- states:  $S = \{s, t, \dots\}$
- sequences of states:  $\rho = s_1 s_2 s_3 \dots \in S^\omega$
- **no distinction between set and its characterizing vector**  
example: if  $AP = \{a_1, a_2, a_3\}$  then  $w \in (2^{AP})^\omega$  is sequence of sets or equivalently, sequence of bitvectors

$$w = \{a_1, a_2\} \emptyset \{a_1, a_3\} \dots = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \dots$$

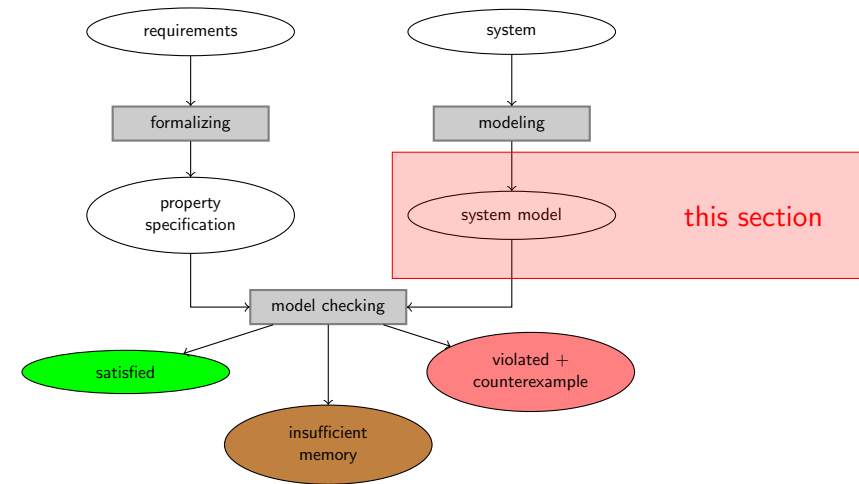
## Outline

- Notations
- Transition systems
- Model Checking of Linear Time Properties
- Regular Languages
  - Finite Automata
  - Büchi Automata
  - Generalized Büchi Automata

## Transition systems

- one way to describe the behaviour of systems
- digraphs where nodes represent **states**, and edges model **transitions**
- **state**:
  - the current phase of a traffic light
  - the current values of all program variables + the program counter
- **transition**: (“state change”)
  - a switch from one phase to the next one
  - the execution of a program statement

## Model checking overview



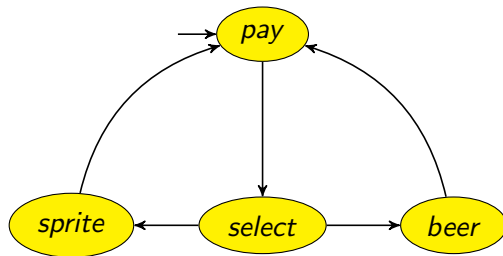
## Transition system

a **transition system**  $TS$  is a tuple  $(S, \rightarrow, I, AP, L)$  where

- $S$  is a set of **states**
- $\rightarrow \subseteq S \times S$  is a **transition relation**
- $I \subseteq S$  is a set of **initial states**
- $AP$  is a set of **atomic propositions**
- $L : S \rightarrow 2^{AP}$  is a **labeling function**

notation:  $s \rightarrow s'$  instead of  $(s, s') \in \rightarrow$

## A beverage vending machine



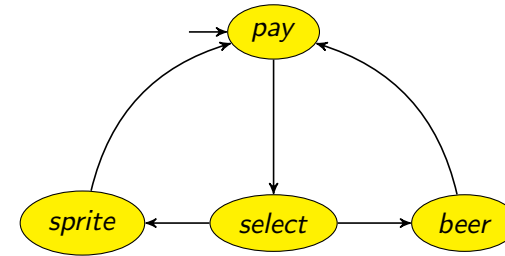
states?, transitions?, initial states?

$TS = (S = \{pay, select, sprite, beer\}, \rightarrow, I = \{pay\}, AP, L)$

where  $\rightarrow$  is the following set of transitions

$pay \rightarrow select$   
 $select \rightarrow beer$                        $select \rightarrow sprite$   
 $beer \rightarrow pay$                           $sprite \rightarrow pay$

## Atomic propositions?



(paid, beer paid, sprite paid will be displayed)

$AP = \{\text{paid, sprite, beer}\}$

$L(\text{pay}) = \emptyset$

$L(\text{select}) = \{\text{paid}\}$

$L(\text{beer}) = \{\text{beer, paid}\}$

$L(\text{sprite}) = \{\text{sprite, paid}\}$

## The role of nondeterminism

here: **nondeterminism is a feature!**

- to model **concurrency by interleaving**
  - no assumption about the relative speed of processes
- to model **implementation freedom**
  - only describes what a system should do, not **how**
- to model **under-specified** systems, or **abstractions** of real systems
  - use incomplete information

in automata theory, nondeterminism may be exponentially more succinct, but that's not the issue here!

## Executions

- execution**  $\varrho$  of  $TS$ : sequence of states

$$\varrho = s_0 s_1 \dots s_n \dots$$

such that

- $s_i \rightarrow s_{i+1}$  for all  $0 \leq i \in \mathbb{N}$
- $s_0 \in I$

(w.l.o.g. consider only infinite executions)

- trace** of an execution: sequence of sets of atomic propositions, i.e.,  $\text{trace}(\varrho) \in (2^{AP})^\omega$

$$\text{trace}(\varrho) = L(s_0) L(s_1) L(s_2) L(s_3) \dots$$

- Traces**( $TS$ ): set of all traces of all executions of  $TS$   
it defines the **observable behaviour** of  $TS$

## Example

$\varrho = \text{pay} \rightarrow \text{select} \rightarrow \text{sprite} \rightarrow \text{pay} \rightarrow \text{select} \rightarrow \text{beer} \dots$

$\text{trace}(\varrho) = \emptyset \{ \text{paid} \} \{ \text{paid, sprite} \} \emptyset \{ \text{paid} \} \{ \text{paid, beer} \} \dots$

interesting properties

- without having paid there is no sprite (true)
- we will see beer infinitely often (false)
- after each beer another coin is inserted (not expressible, needs AP coin, paid may contain credit card)

states  $\sim$  implementation

$\Rightarrow$  **properties only speak about atomic propositions, never about states!**  
(we are not interested in implementation, only result must be correct)

## Outline

- Notations
- Transition systems
- Model Checking of Linear Time Properties
- Regular Languages
  - Finite Automata
  - Büchi Automata
  - Generalized Büchi Automata

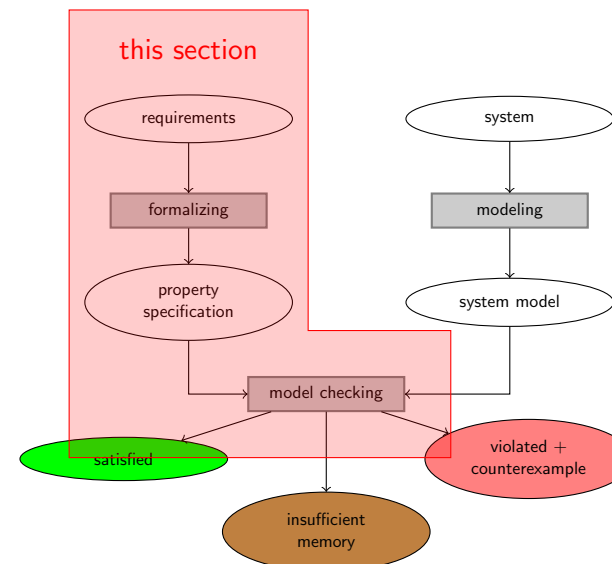
## Summary

transition systems  $\neq$  finite automata since

- there are **no** accept states
- set of states may be countably infinite (but in this lecture: only finite sets of states)
- may have infinite branching
- non-determinism has a different role

$\Rightarrow$  transition systems are appropriate for reactive system behavior

## Model checking overview



## Requirements $\neq$ Specification

### requirements

- high-level description (consider scheduler for exclusive access)
    - (the scheduler should be correct)
    - no two clients get access at the same time
    - the scheduler should be fair
    - there is no deadlock
  - what we observe from system:  $Traces(TS) \subseteq (2^{AP})^\omega$
- $\Rightarrow$  how to answer question “does system satisfy requirements”?  
problem: too imprecise what is fair
- $\Rightarrow$  we need requirements in a precise, i.e., mathematical **specification**

## The requirements of model checking

essentially we need a mechanism to represent the set  $S(R)$  of **allowed traces** for some requirement  $R$  conveniently

- possible classes: finite, regular, context-free, context-sensitive, ...
  - model checking requires checking  $Traces(TS) \subseteq S(R)$   
or equivalently:  $Traces(TS) \cap S(\neg R) = \emptyset$   
where  $\neg R$  describes **forbidden traces**
- $\Rightarrow$  requirements on class of language
- closure under **intersection**
  - **emptiness** decidable
  - expressive enough to represent  $Traces(TS)$  and  $S(\neg R)$
  - use **regular languages**, they are closed under all boolean operations
  - possible representations of regular languages
    - regular expressions
    - non-recursive grammars
    - **finite automata**

## Linear Time Properties

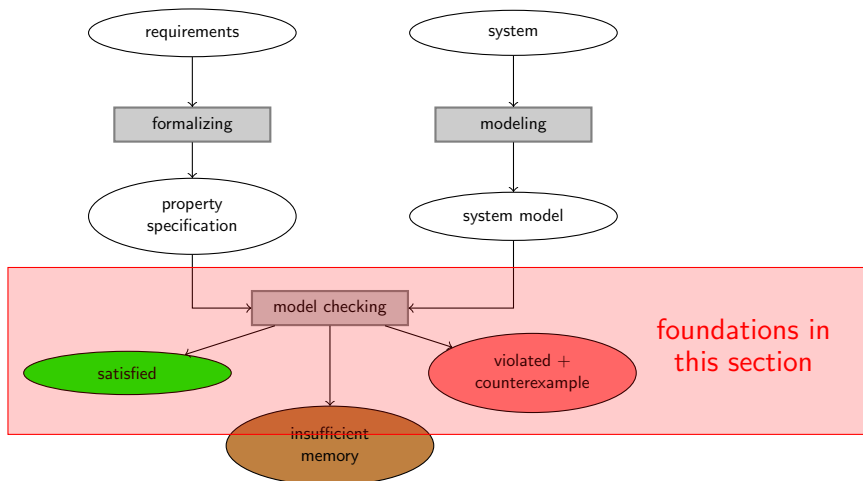
one main idea to specify requirements: describe allowed traces

- specification is set  $S \subseteq (2^{AP})^\omega$  (**linear time property**)
  - system  $TS$  satisfies  $S$  iff every trace of  $TS$  is allowed w.r.t.  $S$ :  
 $Traces(TS) \subseteq S$
  - **model checking of linear time properties**:  
given  $Traces(TS)$  and  $S$ , answer  $Traces(TS) \subseteq S$
- $\Rightarrow$  precise formulation, no ambiguity
- upcoming problems
    - how to **specify sets  $S$  conveniently** ...
    - ... such that  $Traces(TS) \subseteq S$  can be **decided**

## Outline

- Notations
- Transition systems
- Model Checking of Linear Time Properties
- **Regular Languages**
  - Finite Automata
  - Büchi Automata
  - Generalized Büchi Automata

## Model checking overview



## Language of an NFA

- NFA  $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$  and word  $w = A_1 \dots A_n \in \Sigma^*$
- run of  $w$  in  $\mathcal{A}$ : finite sequence  $q_0 q_1 \dots q_n$  such that
  - $q_i \xrightarrow{A_{i+1}} q_{i+1}$  for all  $0 \leq i < n$
- run  $q_0 q_1 \dots q_n$  is **accepting** iff  $q_n \in F$
- $w \in \Sigma^*$  is **accepted** by  $\mathcal{A}$  iff there exists accepting run for  $w$
- **accepted language** of  $\mathcal{A}$ :

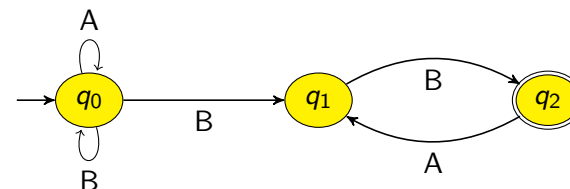
$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \text{there exists an accepting run for } w \text{ in } \mathcal{A} \}$$

- NFA  $\mathcal{A}$  and  $\mathcal{A}'$  are **equivalent** iff  $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$
- language  $\mathcal{L}$  is **regular** iff  $\mathcal{L} = \mathcal{L}(\mathcal{A})$  for some NFA  $\mathcal{A}$

## Finite Automata

a **nondeterministic finite automaton** (NFA)  $\mathcal{A}$  is a tuple  $(Q, \Sigma, \delta, q_0, F)$  where:

- $Q = \{q_0, \dots, q_n\}$  is a **finite set of states**
- $\Sigma$  is an **alphabet**
- $\delta : Q \times \Sigma \rightarrow 2^Q$  is a **transition function**
- $q_0$  is the **initial state**
- $F \subseteq Q$  is a set of **final** (or: **accepting**) states

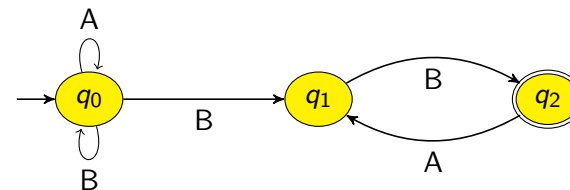


later, after definition:  $\mathcal{L}(\mathcal{A}) = (A \vee B)^* BB(AB)^*$

## Büchi Automata

A **nondeterministic Büchi automaton** (NBA)  $\mathcal{A}$  is a tuple  $(Q, \Sigma, \delta, q_0, F)$  where:

- $Q = \{q_0, \dots, q_n\}$  is a finite set of states
- $\Sigma$  is an **alphabet**
- $\delta : Q \times \Sigma \rightarrow 2^Q$  is a **transition function**
- $q_0 \in Q$  is the initial state
- $F \subseteq Q$  is a set of **final** (or: **accepting**) states



later, after definition:  $\mathcal{L}(\mathcal{A}) = (A \vee B)^* BB(AB)^\omega = \dots BBABABAB \dots$

## Language of a NBA

- NBA  $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$  and word  $w = A_1 \dots A_n \dots \in \Sigma^\omega$
- run for  $w$  in  $\mathcal{A}$  is an infinite sequence  $q_0 q_1 \dots q_n \dots$  such that:
  - $q_i \xrightarrow{A_{i+1}} q_{i+1}$  for all  $i \in \mathbb{N}$
- run  $q_0 q_1 \dots q_n \dots$  is **accepting** iff
  - for infinitely many indices  $i: q_i \in F$
- $w \in \Sigma^\omega$  is **accepted** by  $\mathcal{A}$  iff there exists an accepting run for  $w$
- accepted language of  $\mathcal{A}$ :

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^\omega \mid \text{there exists an accepting run for } w \text{ in } \mathcal{A} \}$$

- NBA  $\mathcal{A}$  and  $\mathcal{A}'$  are **equivalent** iff  $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$
- language  $\mathcal{L}$  is  **$\omega$ -regular** iff  $\mathcal{L} = \mathcal{L}(\mathcal{A})$  for some NBA  $\mathcal{A}$

## GNBAs to specify requirements

Safety properties: (refutation by a finite prefix of an  $\omega$ -word)

- always at most one traffic light is showing green
- green cannot be directly followed by red

Liveness properties: (refutation only by whole  $\omega$ -word)

- we will see green infinitely often
- whenever we select sprite then later on we will get a sprite

- GNBAs are closed under union, intersection, and negation

$\Rightarrow$  many interesting properties can be expressed by GNBAs

## Generalized Büchi Automata

generalized Büchi automaton (GNBA) is tuple

$\mathcal{A} = (Q, \Sigma, \delta, q_0, F_1, \dots, F_k)$  where

- everything is like for NBAs except that
- there are **multiple sets of final states**  $F_1, \dots, F_k$  where each  $F_i \subseteq Q$
- run  $q_0 q_1 \dots q_n \dots$  is **accepting** iff for **each**  $1 \leq j \leq k$  there are **infinitely many indices**  $i: q_i \in F_j$
- NBAs are GNBA's where  $k = 1$
- each GNBA can be translated into equivalent NBA (with states  $Q \times \{1, \dots, \max(1, k)\}$ )

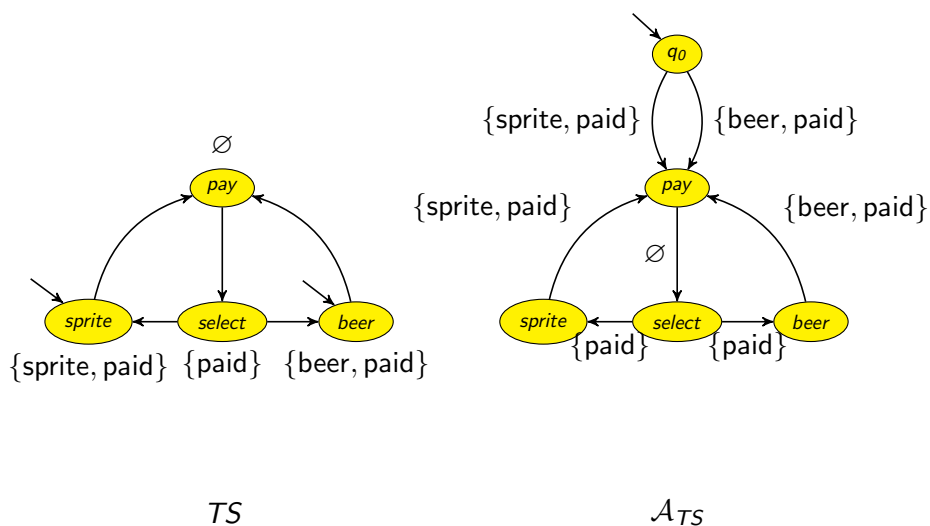
## GNBAs to specify requirements

	$R$	$\neg R$
1.	$\left( \begin{matrix} \text{green1} \\ \text{green2} \end{matrix} \right) \quad k = 0$	
2.	$\left( \begin{matrix} \text{red} \\ \text{green} \end{matrix} \right) \quad k = 0$	
3.	$\left( \begin{matrix} \text{red} \\ \text{green} \end{matrix} \right)$	
4.	$\left( \begin{matrix} \text{select} \\ \text{get} \end{matrix} \right)$	

## Recall: requirements of model checking

- model checking requires checking  $Traces(TS) \cap \mathcal{S}(\neg R) = \emptyset$  where  $\neg R$  describes forbidden traces
- ⇒ requirements on class of language
  - expressive enough to represent  $Traces(TS)$
  - closure under **intersection**
  - **emptiness** decidable

## Example



## Expressing Transition Systems as GNBA

aim: for  $TS = (S, \rightarrow, I, AP, L)$  construct  $\mathcal{A}_{TS}$  with  $\mathcal{L}(\mathcal{A}_{TS}) = Traces(TS)$

problems:

- labels/letters are at the states in  $TS$ , but on the transitions in GNBA
- several initial states in  $TS$ , but only one initial state in GNBA

solution:

- label  $A$  of transition system state corresponds to upcoming letter to read in GNBA
- use states of  $TS$  as states of GNBA, but use new initial state

concrete:  $\mathcal{A}_{TS} = (S \uplus \{q_0\}, 2^{AP}, \delta, q_0)$  with  $\delta$  defined as follows

- $\delta(s, A) = \{s' \mid L(s) = A, s \rightarrow s'\}$
- $\delta(q_0, A) = \bigcup_{s \in I} \delta(s, A)$

## Soundness of $\mathcal{A}_{TS}$

Theorem

$$\mathcal{L}(\mathcal{A}_{TS}) = Traces(TS)$$

let  $w = A_0 A_1 A_2 \dots$

- $w \in Traces(TS)$
- iff exist execution  $s_0 s_1 s_2 \dots$  of  $TS$ :
  - $s_0 \in I$  and for all  $i \in \mathbb{N}$ :
    - $L(s_i) = A_i$
    - $s_i \rightarrow s_{i+1}$
- iff exist (accepting) run  $q_0 s_1 s_2 \dots$  of  $\mathcal{A}_{TS}$ :
  - $s_1 \in \delta(q_0, A_0)$  and for all  $i \geq 1$ :
    - $s_{i+1} \in \delta(s_i, A_i)$
- iff  $w \in \mathcal{L}(\mathcal{A}_{TS})$



## GNBA for Intersection

aim: for  $\mathcal{A}_i = (\mathcal{Q}_i, \Sigma, \delta_i, q_{0,i}, F_{1,i}, \dots, F_{k_i,i})$  construct  $\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2}$  with  $\mathcal{L}(\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$

idea: simulate runs in  $\mathcal{A}_1$  and  $\mathcal{A}_2$  in parallel

- use cartesian product of states
- demand that all final states are visited infinitely often

concrete:  $\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2} = (\mathcal{Q}, \Sigma, \delta, q_0, F'_1, \dots, F'_{k_1}, F''_1, \dots, F''_{k_2})$  where

- $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$
- $q_0 = (q_{0,1}, q_{0,2})$
- $\delta((q_1, q_2), A) = \{(q'_1, q'_2) \mid q'_1 \in \delta_1(q_1, A), q'_2 \in \delta_2(q_2, A)\}$
- $F'_j = F_{j,1} \times \mathcal{Q}_2$  and  $F''_j = \mathcal{Q}_1 \times F_{j,2}$

## Soundness of $\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2}$

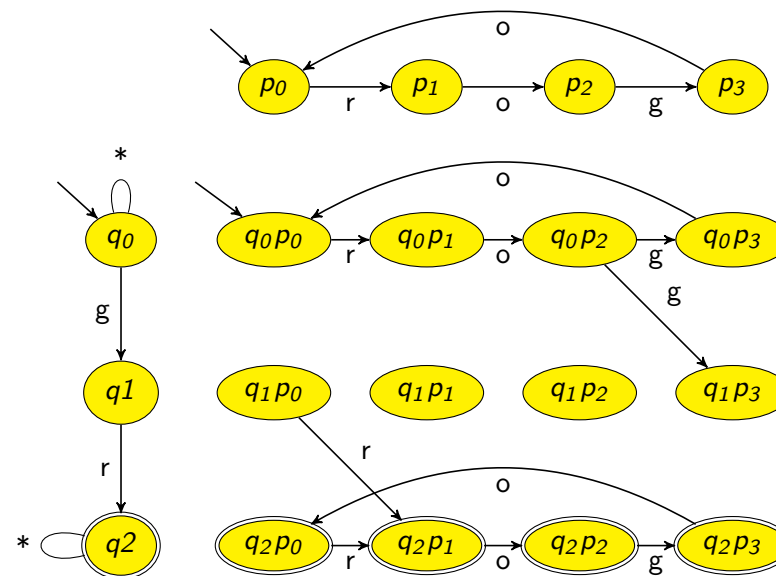
### Theorem

$$\mathcal{L}(\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$$

let  $w = A_0 A_1 A_2 \dots$

- $w \in \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$
- iff exists accepting run  $q_{0,1} q_{1,1} q_{2,1} \dots$  of  $w$  in  $\mathcal{A}_1$  and exists accepting run  $q_{0,2} q_{1,2} q_{2,2} \dots$  of  $w$  in  $\mathcal{A}_2$ :
  - $q_{j+1,i} \in \delta_i(q_{j,i}, A_j)$  for all  $0 \leq j$  and  $1 \leq i \leq 2$
  - for each  $1 \leq j \leq k_i, 1 \leq i \leq 2$  there are infinitely many  $q_{m,i} \in F_{j,i}$
- iff exist accepting run  $(q_{0,1}, q_{0,2}) (q_{1,1}, q_{1,2}) (q_{2,1}, q_{2,2}) \dots$  of  $\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2}$ :
  - $(q_{j+1,1}, q_{j+1,2}) \in \delta((q_{j,1}, q_{j,2}), A_j)$  for all  $0 \leq j$
  - for each  $F'_j$  there are infinitely many  $(q_{m,1}, q) \in F_{j,1} \times \mathcal{Q}_2$  and for each  $F''_j$  there are infinitely many  $(q, q_{m,2}) \in \mathcal{Q}_1 \times F_{j,2}$
- iff  $w \in \mathcal{L}(\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2})$

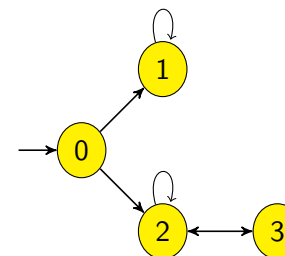
## Example: no direct switch from green to red



## Checking Emptiness of GNBA

let  $G$  be a graph  $(V, E)$  with nodes  $V$  and edges  $E$

- set  $C \subseteq V$  is a **cycle** of  $G$  iff for all  $v_1, v_2 \in C$  there is a non-empty path from  $v_1$  to  $v_2$
- a **strongly connected component (SCC)** is a maximal cycle ( $C$  is SCC iff both  $C$  is cycle and  $C' \supset C$  implies  $C'$  is not a cycle)
- remarks:
  - two SCCs  $C_1$  and  $C_2$  are either disjoint or identical
  - the set of SCCs of a graph can be determined in linear time (Kosaraju)



## Algorithm for Checking $\mathcal{L}(\mathcal{A}) = \emptyset$ for GNBA $\mathcal{A}$

$\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, F_1, \dots, F_k)$  accepts at least one  $\omega$ -word

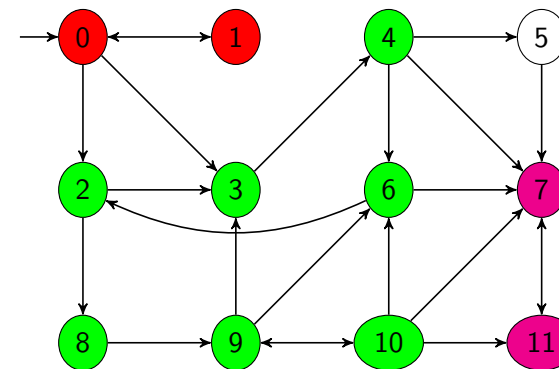
- iff there is accepting run  $q_0 q_1 q_2 \dots$  of  $\mathcal{A}$
- iff there is run  $q_0 q_1 q_2 \dots$  where for each  $F_i$  there is some  $q_{f,i} \in F_i$  that occurs infinitely often
- iff in the graphical representation of  $\mathcal{A}$  there is a path from  $q_0$  to some SCC containing all  $q_{f,i}$ 's
- iff in the graphical representation of  $\mathcal{A}$  there is a path from  $q_0$  to some SCC containing at least one final state of each  $F_i$

$\Rightarrow$  compute SCCs of  $\mathcal{A}$  (linear time, Kosaraju's algorithm)

and perform reachability-analysis from  $q_0$  (linear time, depth first search)

- if no SCC with final states from each  $F_i$  reachable from  $q_0$ :  $\mathcal{L}(\mathcal{A}) = \emptyset$
- otherwise, obtain path from  $q_0$  to  $q_1 \in SCC$  and non-empty path from  $q_1$  to  $q_1$  traversing all nodes in the SCC with corresponding words  $w_{01}$  and  $w_{11}$   
 $\Rightarrow w_{01}w_{11}^\omega \in \mathcal{L}(\mathcal{A})$

## Example (GNBA where input letters are omitted)



$$F_1 = \{0, 9\} \quad F_2 = \{0, 3, 10\} \quad F_3 = \{5, 10, 11\}$$

extracted accepted run:  $0(2346289106)^\omega$

## Summary

- model checking for linear time properties:
  - specify allowed traces  $\mathcal{S}(R)$  or forbidden traces  $\mathcal{S}(\neg R)$
  - decide

$$\text{Traces}(T) \subseteq \mathcal{S}(R) \quad \text{or} \quad \text{Traces}(TS) \cap \mathcal{S}(\neg R) = \emptyset$$

- NFA over finite words  $\rightarrow$  (G)NBA over infinite words  
(regular languages  $\rightarrow$  regular  $\omega$ -languages)
- GNBA can encode several requirements (allowed / forbidden traces)
- GNBA can encode transition systems
- GNBA are closed under Boolean operations (here: only intersection)
- emptiness of GNBA is decidable