Introduction to Model Checking

René Thiemann

Institute of Computer Science
University of Innsbruck

WS 2010/2011

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

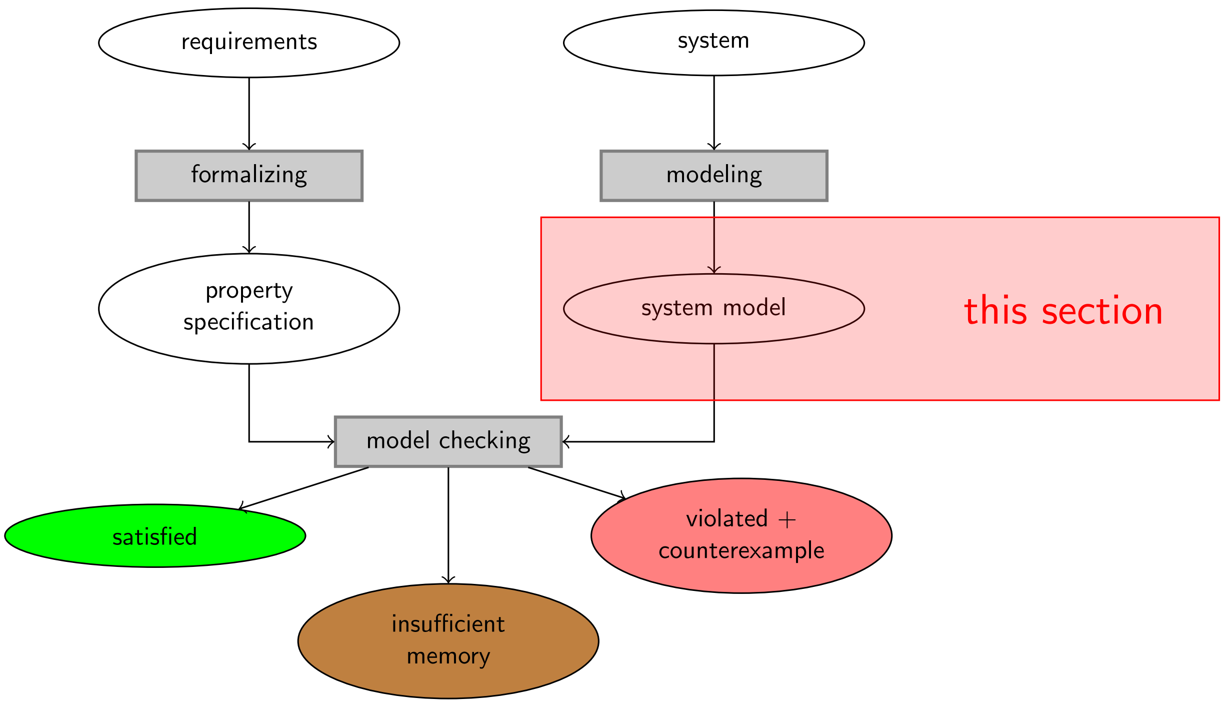
- atomic propositions: $AP = \{a, b, \ldots\}$
- signature: $\Sigma = \{A, B, \ldots\}$, often $\Sigma = 2^{AP}$
- infinite words: $\Sigma^\omega = \{v, w, \ldots\}$ where $w = A_1 A_2 A_3 \ldots$
- states: $S = \{s, t, \ldots\}$
- sequences of states: $\rho = s_1 s_2 s_3 \ldots \in S^\omega$
- no distinction 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

$$w = \{a_1, a_2\} \varnothing \{a_1, a_3\} \ldots = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \ldots$$

Outline

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

Model checking overview



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

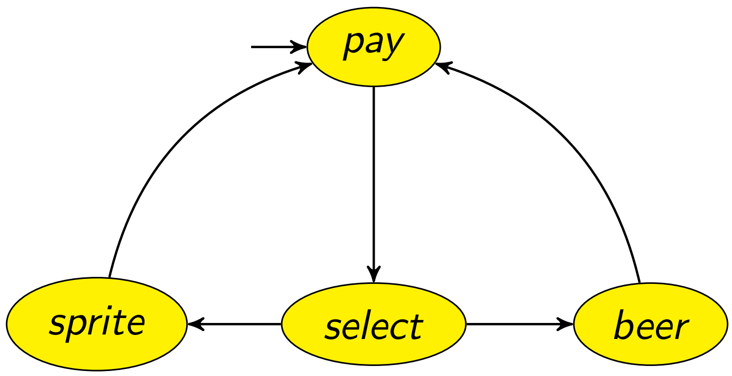
Transition system

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

- S is a set of states
- $\longrightarrow \subseteq S \times S$ 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

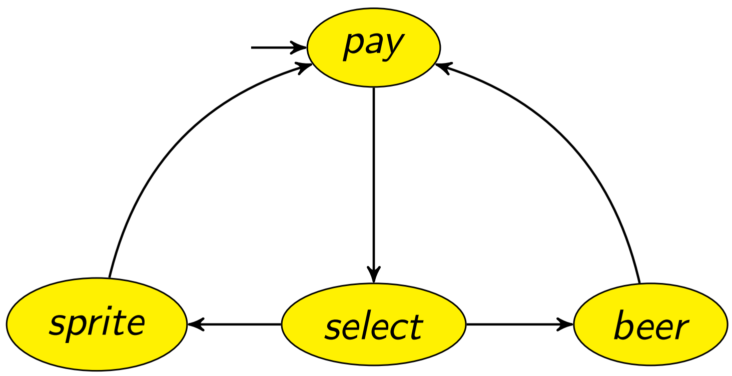
notation: $s \rightarrow s'$ instead of $(s, s') \in \longrightarrow$

A beverage vending machine



states?, transitions?, initial states?

Atomic propositions?



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

Executions

- execution ϱ of TS: sequence of states

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

  such that
  - $s_i \longrightarrow s_{i+1}$ for all $0 \leqslant i \in \mathbb{N}$
  - $s_0 \in I$

  (w.l.o.g. consider only infinite executions)
- trace of an execution: sequence of sets of atomic propositions, i.e., $trace(\varrho) \in (2^{AP})^\omega$

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

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

Example

Summary

transition systems $\neq$ finite automata since

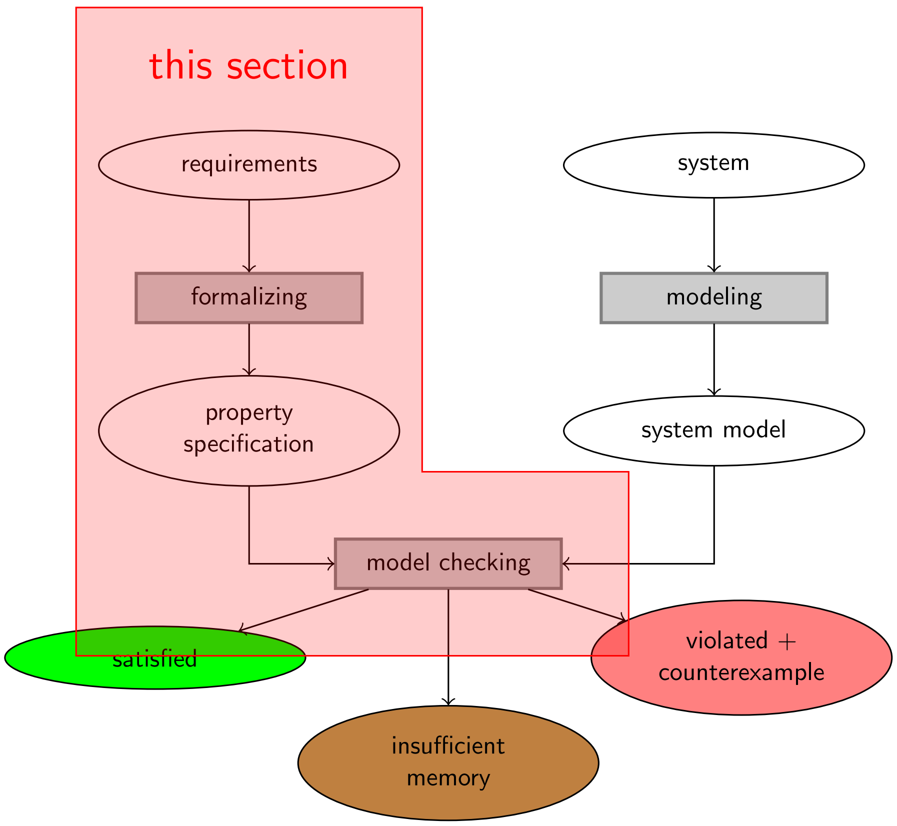
- there are no accept states
- set of states may be countably infinite
  (but in this lecture: only finite sets of states)
- may have infinite branching
- non-determinism has a different role

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

Outline

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

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: $\mathit{Traces}(\mathit{TS}) \subseteq (2^{AP})^\omega$

$\Rightarrow$ how to answer question "does system satisfy requirements"? problem: to imprecise

$\Rightarrow$ we need requirements in a precise, i.e., mathematical specification

Linear Time Properties

one main idea to specify requirements: describe allowed traces

- specification is set $\mathcal{S} \subseteq (2^{AP})^\omega$ (linear time property)
- system TS satisfies $\mathcal{S}$ iff every trace of TS is allowed w.r.t. $\mathcal{S}$:

$$\mathit{Traces}(\mathit{TS}) \subseteq \mathcal{S}$$

- model checking of linear time properties:
  given $\mathit{Traces}(\mathit{TS})$ and $\mathcal{S}$, answer $\mathit{Traces}(\mathit{TS}) \subseteq \mathcal{S}$

$\Rightarrow$ precise formulation, no ambiguity

- upcoming problems
  - how to specify sets $\mathcal{S}$ conveniently ...
  - ...such that $\mathit{Traces}(\mathit{TS}) \subseteq \mathcal{S}$ can be decided

The requirements of model checking

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

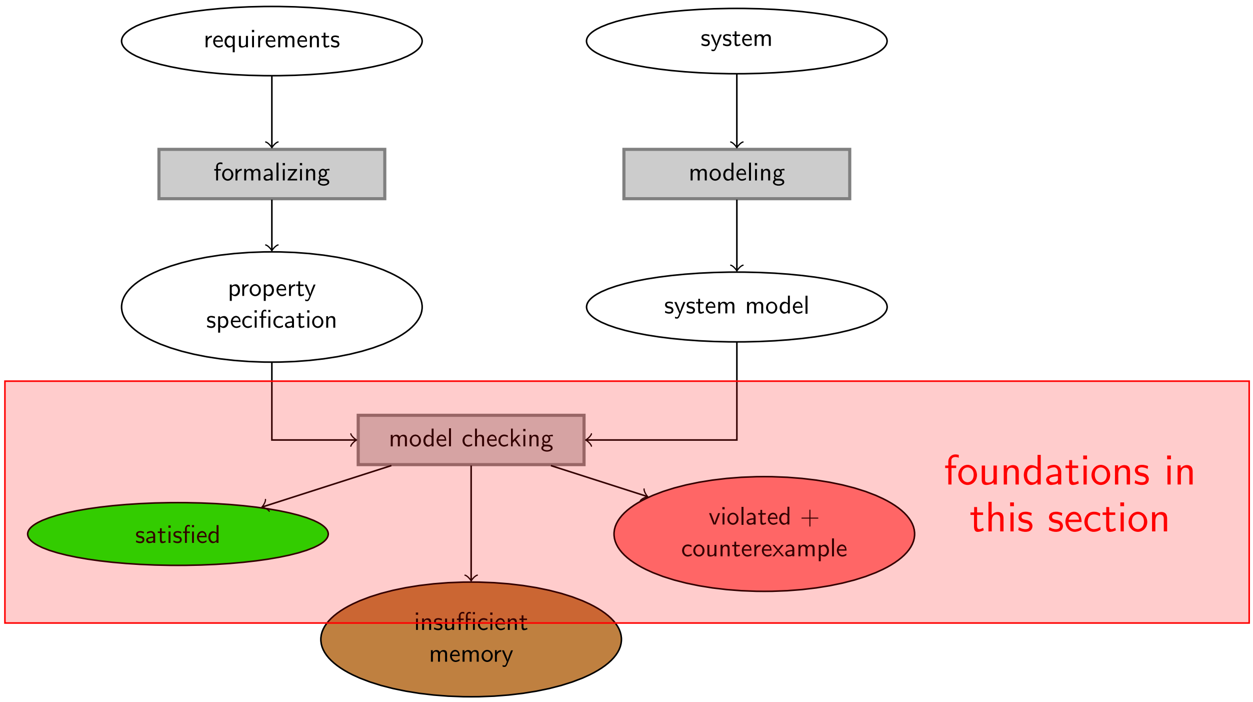
- possible classes: finite, regular, context-free, context-sensitive, ...
- model checking requires checking $\mathit{Traces}(\mathit{TS}) \subseteq \mathcal{S}(R)$
  or equivalently: $\mathit{Traces}(\mathit{TS}) \cap \mathcal{S}(\neg R) = \varnothing$
  where $\neg R$ describes forbidden traces

$\Rightarrow$ requirements on class of language
  - closure under intersection
  - emptyness decidable
  - expressive enough to represent $\mathit{Traces}(\mathit{TS})$ and $\mathcal{S}(\neg R)$

- use regular languages, they are closed under all boolean operations
- possible representations of regular languages
  - regular expressions
  - non-recursive grammars
  - finite automata

Outline

Model checking overview



Finite Automata

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

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

Language of an NFA

- NFA $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$ 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 $0 \leqslant i < n$
- 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 $\mathcal{A}$:

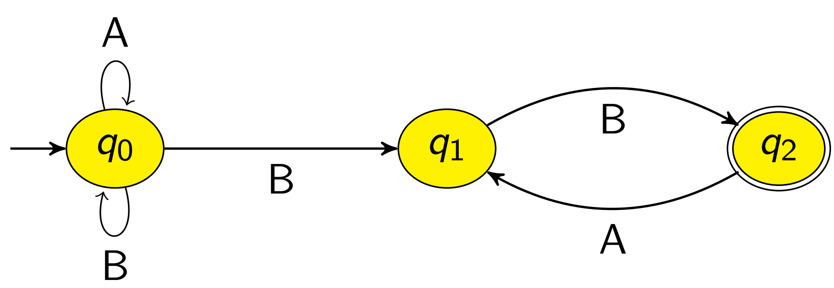
$$\mathcal{L}(\mathcal{A}) = \left\{ w \in \Sigma^* \mid \text{ there exists an accepting run for } w \text{ in } \mathcal{A} \right\}$$

- 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 $\mathcal{A}$

Büchi Automata

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

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



Language of a NBA

- NBA $\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$ 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 ω-regular iff $\mathcal{L} = \mathcal{L}(\mathcal{A})$ for some NBA $\mathcal{A}$

Generalized Büchi Automata

generalized Büchi automaton (GNBA) is tuple
$\mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, F_1, \dots, F_k)$ where

- everything is like for NBAs except that
- there are multiple sets of final states $F_1, \dots, F_k$ where each $F_i \subseteq \mathcal{Q}$
- run $q_0\, q_1 \;\dots\; q_n \;\dots$ is accepting iff for each $1 \leqslant j \leqslant k$ there are

  infinitely many indices i: $q_i \in F_j$
- NBAs are GNBAs where $k = 1$
- each GNBA can be translated into equivalent NBA (with states $\mathcal{Q} \times \{1, \dots, \max(1, k)\}$)

GNBAs to specify requirements

Safety properties: (refutation by a finite prefix of an ω-word)

1. always at most one traffic light is showing green
2. green cannot be directly followed by red

Liveness properties: (refutation only by whole ω-word)

3. we will see green infinitely often
4. 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

GNBAs to specify requirements

	R	$\neg R$
1.		
2.		
3.		
4.		

Recall: requirements of model checking

- model checking requires checking $Traces(TS) \cap \mathcal{S}(\neg R) = \varnothing$
  where $\neg R$ describes forbidden traces

$\Rightarrow$ requirements on class of language
  - expressive enough to represent $Traces(TS)$
  - closure under intersection
  - emptyness decidable

Expressing Transition Systems as GNBAs

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 GNBAs
- several initial states in TS, but only one initial state in GNBAs

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 δ defined as follows

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

Example

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

Theorem

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

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 $F''_j = \mathcal{Q}_1 \times F_{j,2}$

Example: no direct switch from green to red

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

Theorem

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

Checking Emptiness of GNBAs

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 $\mathcal{L}(\mathcal{A}) = \varnothing$ for GNBA $\mathcal{A}$

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

- iff there is accepting run $q_0\, q_1\, q_2\, \ldots$ of $\mathcal{A}$
- iff there is run $q_0\, q_1\, q_2\, \ldots$ 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: $\mathcal{L}(\mathcal{A}) = \varnothing$
- 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)

Summary

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

$$\mathit{Traces}(T) \subseteq \mathcal{S}(R) \qquad \text{or} \qquad \mathit{Traces}(TS) \cap \mathcal{S}(\neg R) = \varnothing$$

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