A Quick Tour on LTL Model Checking

Lecture #1 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: katoen@cs.rwth-aachen.de

October 22, 2006

Course topics

- Linear-time and computation tree temporal logic (summary)
  - syntax, semantics, model-checking algorithms, expressiveness
- Equivalences and abstraction
  - bisimulation, simulation, minimisation algorithms
  - stutter-bisimulation, stutter trace-equivalence, divergence
- Partial-order reduction
  - independence, ample set method, branching-time POR

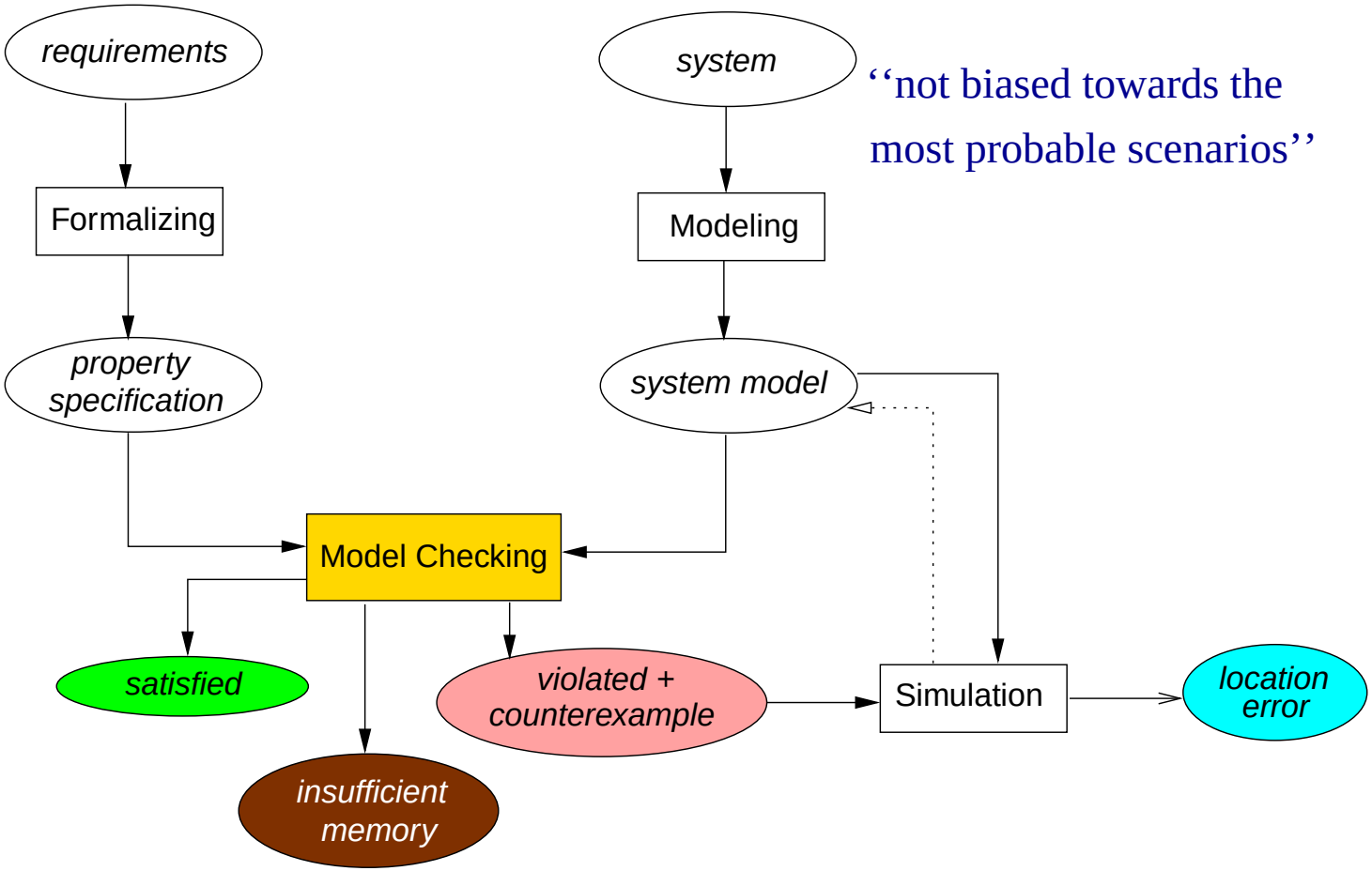
Course topics

- Reduced binary decision diagrams
  - Boolean functions, operations, CTL model checking with ROBDDs
- Timed automata
  - semantics, region equivalence, timed reachability, zone automata, DBMs
- Probabilistic model checking
  - Markov chains, probabilistic CTL, model-checking algorithms

Course organization

- Lectures: twice per week (AH4/AH1, Mon + Thu; check web-page!)
- Exercises: once per week (AH1, Fri)
  - marked exercises (50% of points needed + one example on board)
  - assistent: Tingting Han and Ivan Zapreev
- Course material:
  - draft book “Principles of Model Checking” (Baier & Katoen)
  - find flaws? please report them on WIKI-page
  - one set of exercises waived if you find most flaws in some chapter
- Exam: exam at end of WS 2006/07

Model checking overview



Transition system

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

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

S and *Act* are either finite or countably infinite

Notation: $s \xrightarrow{\alpha} s'$ instead of $(s, \alpha, s') \in \longrightarrow$

Paths

- An *infinite path fragment* π is an infinite state sequence:

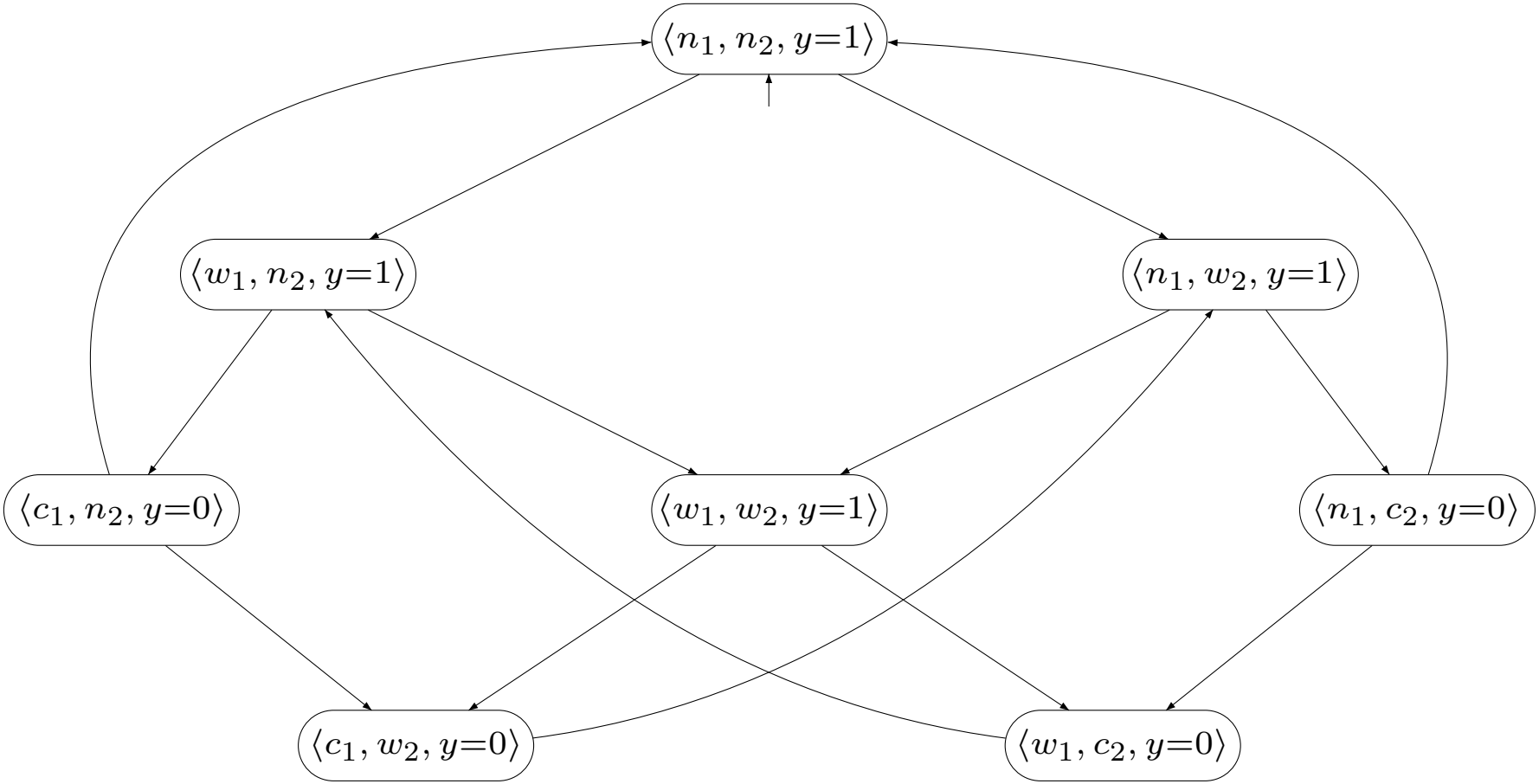
$$s_0\, s_1\, s_2 \ldots \quad \text{such that } s_i \in \mathit{Post}(s_{i-1}) \text{ for all } i > 0$$

- Notations for path fragment $\pi = s_0\, s_1\, s_2 \ldots$:
  - $\mathit{first}(\pi) = s_0 = \pi[0]$; let $\pi[j] = s_j$ denote the j-th state of π
  - j-th prefix $\pi[..j] = s_0\, s_1 \ldots s_j$ and j-th suffix $\pi[j..] = s_j\, s_{j+1} \ldots$

- A *path* of *TS* is an initial, maximal path fragment
  - a *maximal* path fragment is either finite ending in a terminal state, or infinite
  - a path fragment is *initial* if $s_0 \in I$

- $\mathit{Paths}(s)$ is the set of maximal path fragments π with $\mathit{first}(\pi) = s$

A mutual exclusion algorithm



Traces

- Actions are mainly used to model the (possibility of) interaction
  - synchronous or asynchronous communication
- Here, focus on the states that are visited during executions
  - the states themselves are not “observable”, but just their atomic propositions
- Consider sequences of the form $L(s_0)\,L(s_1)\,L(s_2)\ldots$
  - just register the (set of) atomic propositions that are valid along the execution
  - instead of execution $s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \ldots$

  $\Rightarrow$ this is called a *trace*
- For a transition system without terminal states:
  - traces are infinite words over the alphabet 2^{AP}, i.e., they are in $\left(2^{AP}\right)^{\omega}$

Traces

- Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system without terminal states
- The *trace* of path fragment $\pi = s_0\, s_1 \ldots$ is $trace(\pi) = L(s_0)\, L(s_1) \ldots$
  - the trace of $\widehat{\pi} = s_0\, s_1 \ldots s_n$ is $trace(\widehat{\pi}) = L(s_0)\, L(s_1) \ldots L(s_n)$.
- The set of traces of a set Π of paths: $trace(\Pi) = \{\, trace(\pi) \mid \pi \in \Pi \,\}$
- $Traces(s) = trace(Paths(s))$ $\qquad$ $Traces(TS) = \bigcup_{s \in I} Traces(s)$
- $Traces_{fin}(s) = trace(Paths_{fin}(s))$ $\qquad$ $Traces_{fin}(TS) = \bigcup_{s \in I} Traces_{fin}(s)$

Linear-time properties

- Linear-time properties specify the traces that a TS must exhibit
  - LT-property specifies the admissible behaviour of the system
  - later, a logical formalism will be introduced for specifying LT properties

- A *linear-time property* (LT property) over *AP* is a subset of $\left(2^{AP}\right)^{\omega}$
  - finite words are not needed, as it is assumed that there are no terminal states

- *TS* (over *AP*) *satisfies* LT-property P (over *AP*):

$$TS \models P \quad \text{if and only if} \quad Traces(TS) \subseteq P$$

  - *TS* satisfies the LT property P if all its "observable" behaviors are admissible

Linear temporal logic

modal logic over infinite sequences [Pnueli 1977]

- Propositional logic
  - $a \in AP$ — atomic proposition
  - $\neg\phi$ and $\phi \wedge \psi$ — negation and conjunction
- Temporal operators
  - $\bigcirc \phi$ — neXt state fulfills ϕ
  - ϕ U ψ — ϕ holds Until a ψ-state is reached
- Auxiliary temporal operators
  - $\Diamond \phi \equiv$ true U ϕ — eventually ϕ
  - $\Box \phi \equiv \neg \Diamond \neg \phi$ — always ϕ

Derived operators

$$
\begin{aligned}
lcl\phi \ \wedge \ \psi &\equiv \neg(\neg\phi \ \vee \ \neg\psi) \\
\phi \ \Rightarrow \ \psi &\equiv \neg\phi \ \vee \ \psi \\
\phi \Leftrightarrow \psi &\equiv (\phi \Rightarrow \psi) \ \wedge \ (\psi \Rightarrow \phi) \\
\phi \oplus \psi &\equiv (\phi \ \wedge \ \neg\psi) \ \vee \ (\neg\phi \ \wedge \ \psi) \\
\text{true} &\equiv \phi \ \vee \ \neg\phi \\
\text{false} &\equiv \neg\,\text{true} \\
\Diamond\,\phi &\equiv \text{true}\,\mathsf{U}\,\phi \qquad \text{“sometimes in the future”} \\
\Box\,\phi &\equiv \neg\Diamond\,\neg\phi \qquad \text{“from now on for ever”}
\end{aligned}
$$

Example properties expressed in LTL

Semantics over words

The LT-property induced by LTL formula φ over *AP* is:

$\mathit{Words}(\varphi) = \left\{\sigma \in \left(2^{AP}\right)^{\omega} \mid \sigma \models \varphi\right\}$, where $\models$ is the smallest relation satisfying:

$$
\begin{array}{lcll}
\sigma & \models & \text{true} & \\
\sigma & \models & a & \text{iff} \quad a \in A_0 \quad (\text{i.e., } A_0 \models a) \\
\sigma & \models & \varphi_1 \wedge \varphi_2 & \text{iff} \quad \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 \\
\sigma & \models & \neg\varphi & \text{iff} \quad \sigma \not\models \varphi \\
\sigma & \models & \bigcirc \varphi & \text{iff} \quad \sigma[1..] = A_1 A_2 A_3 \ldots \models \varphi \\
\sigma & \models & \varphi_1 \,\mathsf{U}\, \varphi_2 & \text{iff} \quad \exists j \geqslant 0.\ \sigma[j..] \models \varphi_2 \text{ and } \sigma[i..] \models \varphi_1,\ 0 \leqslant i < j
\end{array}
$$

Semantics over paths and states

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system without terminal states, and let φ be an LTL-formula over AP.

- For infinite path fragment π of TS:

$$\pi \models \varphi \qquad \text{iff} \qquad trace(\pi) \models \varphi$$

- For state $s \in S$:

$$s \models \varphi \qquad \text{iff} \qquad \forall \pi \in Paths(s).\ \pi \models \varphi$$

- TS satisfies φ, denoted $TS \models \varphi$, if $Traces(TS) \subseteq Words(\varphi)$

Semantics for transition systems

$TS \models \varphi$

iff (* transition system semantics *)

$Traces(TS) \subseteq Words(\varphi)$

iff (* definition of $\models$ for LT-properties *)

$TS \models Words(\varphi)$

iff (* Definition of $Words(\varphi)$ *)

$\pi \models \varphi$ for all $\pi \in Paths(TS)$

iff (* semantics of $\models$ for states *)

$s_0 \models \varphi$ for all $s_0 \in I$.

Equivalence

LTL formulas ϕ, ψ are *equivalent*, denoted $\phi \equiv \psi$, if:

$$\mathit{Words}(\phi) = \mathit{Words}(\psi)$$

Important equivalences

Duality:

$$\neg \Box \phi \equiv \Diamond \neg \phi$$
$$\neg \Diamond \phi \equiv \Box \neg \phi$$
$$\neg \bigcirc \phi \equiv \bigcirc \neg \phi$$

Idempotency:

$$\Box \Box \phi \equiv \Box \phi$$
$$\Diamond \Diamond \phi \equiv \Diamond \phi$$
$$\phi \,\mathsf{U}\, (\phi \,\mathsf{U}\, \psi) \equiv \phi \,\mathsf{U}\, \psi$$

Absorption:

$$\Diamond \Box \Diamond \phi \equiv \Box \Diamond \phi$$
$$\Box \Diamond \Box \phi \equiv \Diamond \Box \phi$$

Distribution:

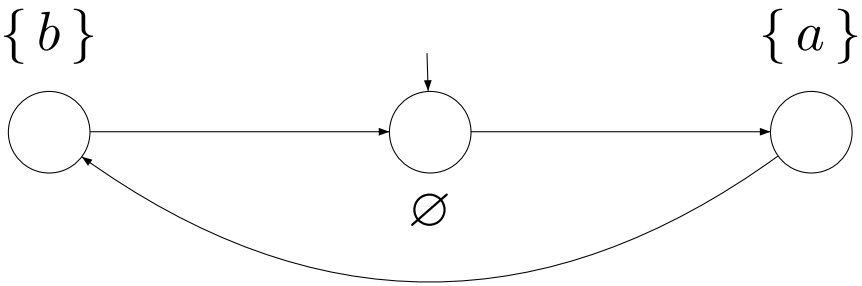
$$\bigcirc (\phi \,\mathsf{U}\, \psi) \equiv (\bigcirc \phi) \,\mathsf{U}\, (\bigcirc \psi)$$

Expansion:

$$\phi \,\mathsf{U}\, \psi \equiv \psi \vee (\phi \wedge \bigcirc (\phi \,\mathsf{U}\, \psi))$$
$$\Diamond \phi \equiv \phi \vee \bigcirc \Diamond \phi$$
$$\Box \phi \equiv \phi \wedge \bigcirc \Box \phi$$

Distributive laws

$\Diamond(a \wedge b) \not\equiv \Diamond a \wedge \Diamond b$ and $\Box(a \vee b) \not\equiv \Box a \vee \Box b$



LTL model-checking problem

Given finite transition system TS and LTL-formula φ:

yields “yes” if $TS \models \varphi$, and “no” (plus a counterexample) if $TS \not\models \varphi$

Automata for LTL formulas

Büchi automata

A *nondeterministic Büchi automaton* (NBA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, Q_0, F)$ where:

- Q is a finite set of states with $Q_0 \subseteq Q$ a set of initial states
- Σ is an alphabet
- $\delta : Q \times \Sigma \to 2^Q$ is a transition function
- $F \subseteq Q$ is a set of accept (or: final) states

The size of $\mathcal{A}$, denoted $|\mathcal{A}|$, is the number of states and transitions in $\mathcal{A}$:

$$|\mathcal{A}| \;=\; |Q| + \sum_{q \in Q} \sum_{A \in \Sigma} |\, \delta(q, A) \,|$$

NBA and ω-regular languages

The class of languages accepted by NBA

agrees with the class of ω-regular languages

(1) any ω-regular language is recognized by an NBA

(2) for any NBA $\mathcal{A}$, the language $\mathcal{L}_\omega(\mathcal{A})$ is ω-regular

ω-regular expressions

1. $\underline{\varnothing}$ and $\underline{\varepsilon}$ are regular expressions over Σ

2. if $A \in \Sigma$ then $\underline{A}$ is a regular expression over Σ

3. if E, E_1 and E_2 are regular expressions over Σ
   then so are $E_1 + E_2$, $E_1.E_2$ and E^*

E^+ is an abbreviation for the regular expression $E.E^*$

An ω-*regular expression* G over Σ has the form:

$$G = E_1.F_1^\omega + \ldots + E_n.F_n^\omega \quad \text{for } n > 0$$

where E_i, F_i are regular expressions over Σ such that $\varepsilon \notin \mathcal{L}(F_i)$, for all $0 < i \leqslant n$

NBA are more expressive than DBA

For finite automata, NFA and DFA are equally expressive

but for NBA this is no longer true:

There is no DBA that accepts $\mathcal{L}_\omega((A+B)^*B^\omega)$

Observation

$TS \models \varphi$ if and only if $\mathit{Traces}(TS) \subseteq \mathit{Words}(\varphi)$

if and only if $\mathit{Traces}(TS) \cap \big((2^{AP})^\omega \setminus \mathit{Words}(\varphi)\big) = \varnothing$

if and only if $\mathit{Traces}(TS) \cap \underbrace{\mathit{Words}(\neg\varphi)}_{\mathcal{L}_\omega(\mathcal{A}_{\neg\varphi})} = \varnothing$

if and only if $TS \otimes \mathcal{A}_{\neg\varphi} \models \Diamond\Box \underbrace{\bigwedge_{q \in F} \neg q}_{\neg F}$

LTL model checking is reduced to checking whether an accept state is visited in $TS \otimes \mathcal{A}_{\neg\varphi}$ infinitely often

Synchronous product

For transition system $TS = (S, Act, \rightarrow, I, AP, L)$ without terminal states and $\mathcal{A} = (Q, \Sigma, \delta, Q_0, F)$ an NBA with $\Sigma = 2^{AP}$ and $Q_0 \cap F = \varnothing$, let:

$$TS \otimes \mathcal{A} \;=\; (S', Act, \rightarrow', I', AP', L') \qquad \text{where}$$

- $S' = S \times Q$, $AP' = Q$ and $L'(\langle s, q \rangle) \;=\; \{\, q \,\}$
- $\rightarrow'$ is the smallest relation defined by: $\dfrac{s \xrightarrow{\alpha} t \;\wedge\; q \xrightarrow{L(t)} p}{\langle s, q \rangle \xrightarrow{\alpha}{}' \langle t, p \rangle}$
- $I' = \{\, \langle s_0, q \rangle \mid s_0 \in I \;\wedge\; \exists q_0 \in Q_0.\; q_0 \xrightarrow{L(s_0)} q \,\}$

without loss of generality it may be assumed that $TS \otimes \mathcal{A}$ has no terminal states

Cycle detection

Let *TS* be a finite transition system without terminal states over *AP*,

$$TS \not\models \Diamond\Box\Phi$$

if and only if

$$\exists s \in \mathit{Reach}(TS).\, s \not\models \Phi \ \wedge\ s \text{ is on a cycle in } TS$$

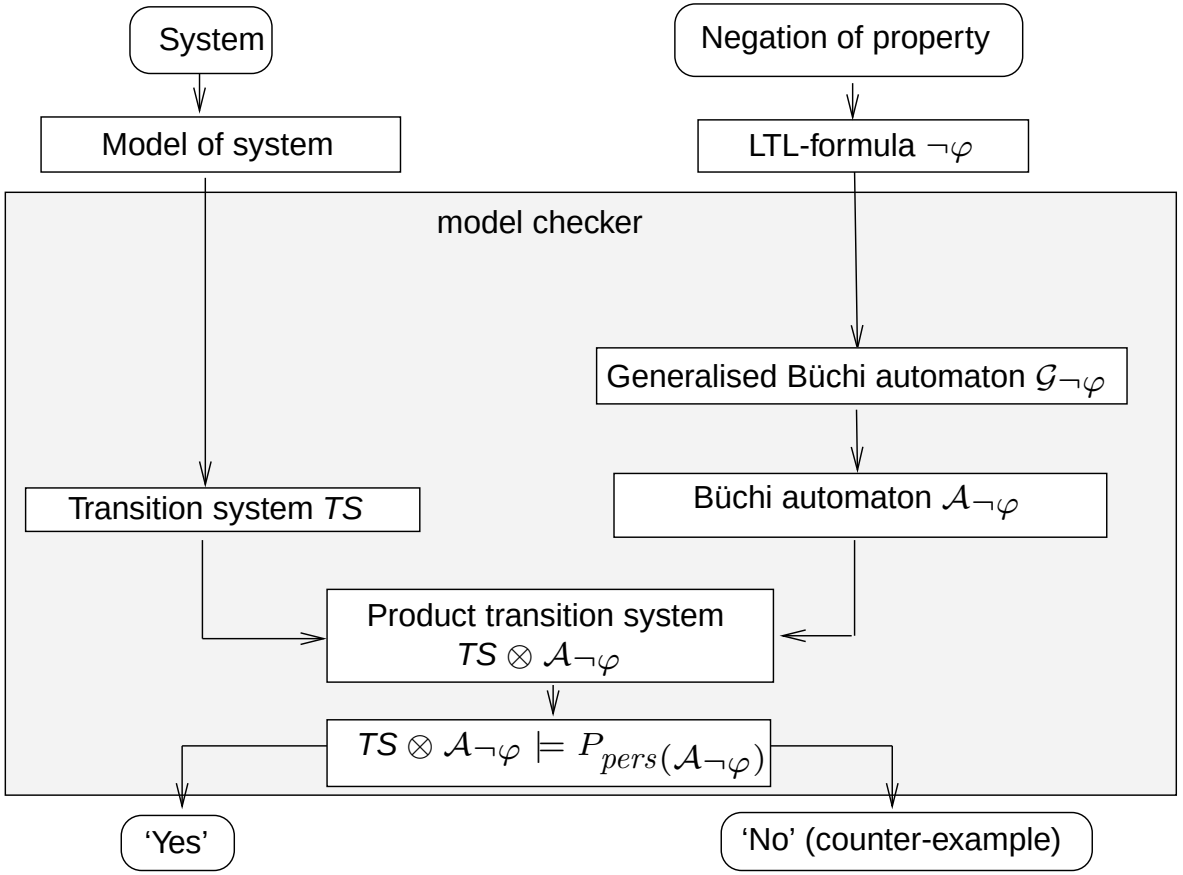
Nested depth-first search

- Idea: perform the two depth-first searches in an *interleaved* way
  - the outer DFS serves to encounter all reachable $\neg\Phi$-states
  - the inner DFS seeks for backward edges leading to a $\neg\Phi$-state

- *Nested DFS*
  - on full expansion of $\neg\Phi$-state s in the outer DFS, start inner DFS
  - in inner DFS, visit all states reachable from s that are *unvisited* in the inner DFS so far
  - no backward edge to s? continue the outer DFS (look for next $\neg\Phi$ state)

- *Counterexample generation*: DFS stack concatenation
  - stack U for the outer DFS = path fragment from $s_0 \in I$ to s (in reversed order)
  - stack V for the inner DFS = a cycle from state s to s (in reversed order)

Overview of LTL model checking



Main result

[Vardi, Wolper & Sistla 1986]

For any LTL-formula φ (over AP) there exists an NBA $\mathcal{A}_\varphi$ over 2^{AP} such that:

(a) $\mathit{Words}(\varphi) = \mathcal{L}_\omega(\mathcal{A}_\varphi)$

(b) $\mathcal{A}_\varphi$ can be constructed in time and space $\mathcal{O}\left(|\varphi| \cdot 2^{|\varphi|}\right)$

$\Rightarrow$ *every LTL-formula expresses an ω-regular property!*

NBA are more expressive than LTL

There is no LTL formula φ with *Words*$(\varphi) = P$ for the LT-property:

$$P = \left\{ A_0 A_1 A_2 \ldots \in \left(2^{\{a\}}\right)^\omega \mid a \in A_{2i} \text{ for } i \geqslant 0 \right\}$$

But there exists an NBA $\mathcal{A}$ with $\mathcal{L}_\omega(\mathcal{A}) = P$

$\Rightarrow$ *there are ω-regular properties that cannot be expressed in LTL!*

Complexity for LTL model checking

The time and space complexity of LTL model checking is in $\mathcal{O}\left(|TS| \cdot 2^{|\varphi|}\right)$

On-the-fly LTL model checking

- Idea: find a counter-example *during* the generation of $\mathit{Reach}(\mathit{TS})$ and $\mathcal{A}_{\neg\varphi}$
  - exploit the fact that $\mathit{Reach}(\mathit{TS})$ and $\mathcal{A}_{\neg\varphi}$ can be generated in parallel

$\Rightarrow$ Generate $\mathit{Reach}(\mathit{TS} \otimes \mathcal{A}_{\neg\varphi})$ "on demand"

- consider a new vertex only if no accepting cycle has been found yet
- only consider the successors of a state in $\mathcal{A}_{\neg\varphi}$ that match current state in *TS*

$\Rightarrow$ Possible to find an accepting cycle without generating $\mathcal{A}_{\neg\varphi}$ entirely

- This on-the-fly scheme is adopted in e.g. the model checker SPIN