Introduction

Lecture #1 of Advanced Model Checking

Joost-Pieter Katoen

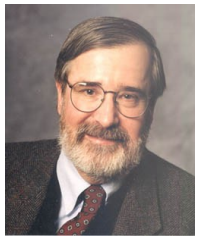
Lehrstuhl 2: Software Modeling & Verification

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

April 15, 2009

Model checking

- Automated model-based verification and debugging technique
  - model of system = Kripke structure ≈ labeled transition system
  - properties expressed in temporal logic like LTL or CTL
  - provides counterexamples in case of property refutation
- Various striking examples
  - Needham-Schroeder security protocol, storm surge barrier, C code
- 2008: Pioneers awarded prestigious ACM Turing Award



Course topics

- Abstraction
  - bisimulation, simulation, minimization algorithms
  - stutter-bisimulation, stutter trace-equivalence, divergence
  - preservation of temporal logical formulae
- Partial-order reduction
  - independence, ample set method, branching-time POR

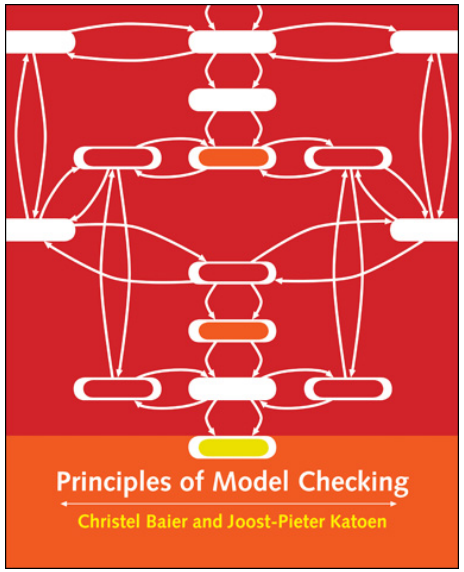
Course topics

- 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 (AH6/5056, Wed + Fri; check web-page!)
- Exercises: once per week (AH2, Mon, start: April 27)
  - marked exercises (50% of points needed + one example on board)
  - assistent: Alexandru Mereacre and Haidi Yue
- Course material:
  - book “Principles of Model Checking” (Baier & Katoen)
  - several copies are available in CS library
  - coverage: chapters 6.7, 7, 8, 9, and 10
- Exam: week 32 and (repetition in) week 39

Principles of Model Checking



CHRISTEL BAIER

TU Dresden, Germany

JOOST-PIETER KATOEN

RWTH Aachen University, Germany

"This book offers one of the most comprehensive introductions to logic model checking techniques available today. The authors have found a way to explain both basic concepts and foundational theory thoroughly and in crystal clear prose. Highly recommended for anyone who wants to learn about this important new field, or brush up on their knowledge of the current state of the art."

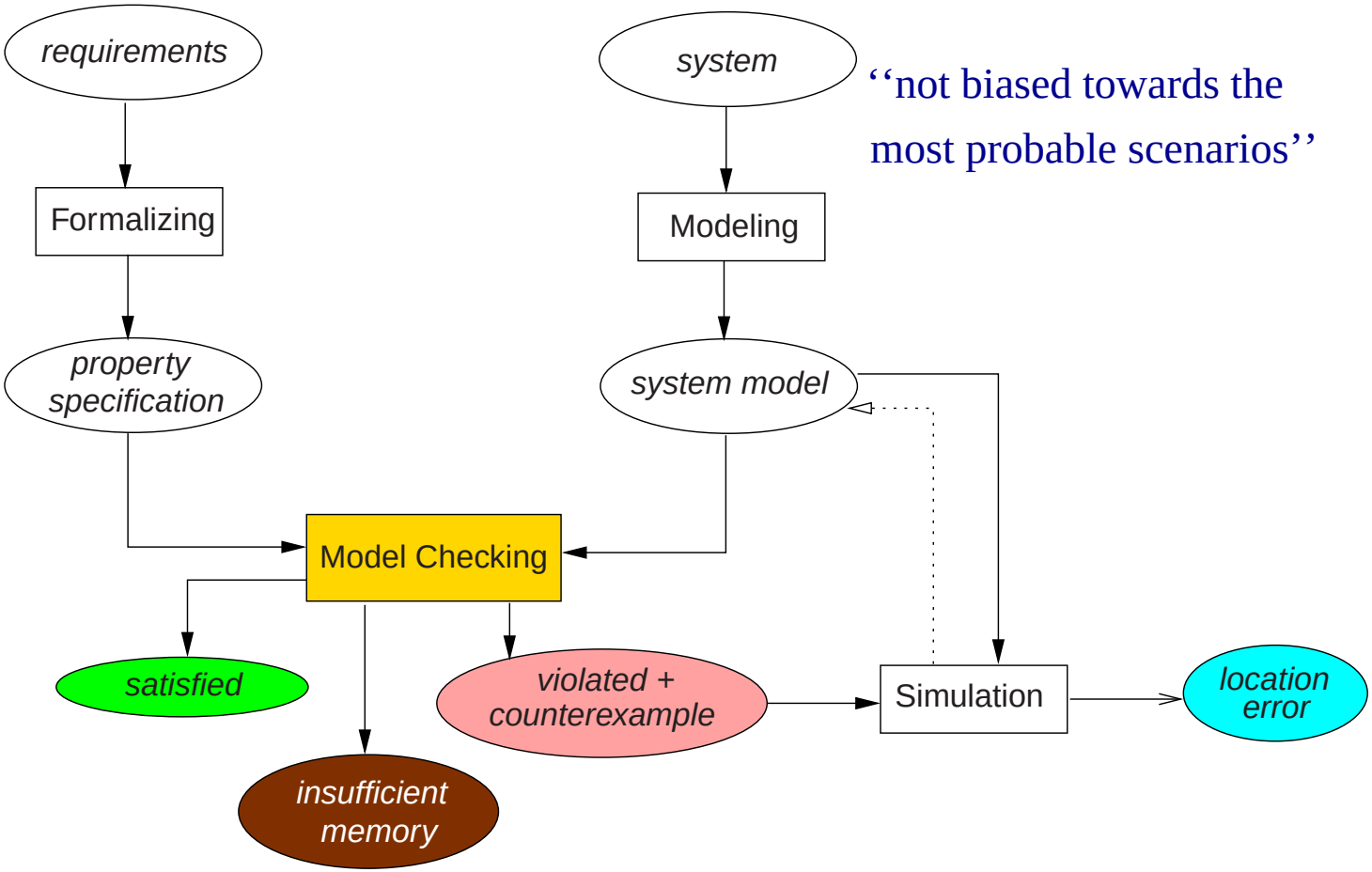
(**Gerard J. Holzmann, NASA JPL, Pasadena**)

Milestones in formal verification

- Mathematical approach towards program correctness (Turing, 1949)
- Syntax-based technique for sequential programs (Hoare, 1969)
  - for a given input, does a computer program generate the correct output?
  - based on compositional proof rules expressed in predicate logic
- Syntax-based technique for concurrent programs (Pnueli, 1977)
  - can handle properties referring to situations during the computation
  - based on proof rules expressed in temporal logic
- Automated verification of concurrent programs (Emerson & Clarke, 1981)
  - model-based instead of proof-rule based approach
  - does the concurrent program satisfy a given (logical) property?

these formal techniques are not biased towards the most probable scenarios

Model checking overview



Models := transition systems

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 cannot be prolonged
  - a path fragment is *initial* if $s_0 \in I$

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

Example

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
- Traces are 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
- For transition systems without terminal states:
  - traces are infinite words over the alphabet 2^{AP}, i.e., they are in $\left(2^{AP}\right)^{\omega}$
  - we will (mostly) assume that there are no terminal states

Traces

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

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

LTL: a logic for LT properties

[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

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

Trace equivalence, LT properties, and LTL

For TS and TS' be finite transition systems (over AP):

$$Traces(TS) = Traces(TS')$$

if and only if

$$(\forall \text{LT property } P.\ TS \models P \text{ iff } TS' \models P)$$

$$Traces(TS) = Traces(TS')$$

implies

$$(\forall \text{LTL formula } \varphi.\ TS \models \varphi \text{ iff } TS' \models \varphi)$$

Model-checking time complexities

- Let *TS* be a transition system with N states and M transitions
- Model-checking LTL-formula Φ has time-complexity $\mathcal{O}((N+M)\cdot 2^{|\Phi|})$
  - linear in the size of *TS* and exponential in the size of Φ
- Model-checking CTL-formula Φ has time-complexity $\mathcal{O}((N+M)\cdot|\Phi|)$
  - linear in the size of *TS* and in the size of Φ
- Can *TS* be made smaller prior or during model checking?
  - Yes. This is the main idea behind abstraction.

Abstraction

Reduce (a huge) TS to (a small) $\widehat{TS}$ prior or during model checking

Relevant issues:

- What is the formal relationship between TS and $\widehat{TS}$?
- Can $\widehat{TS}$ be obtained algorithmically and efficiently?
- Which logical fragment (of LTL, CTL, CTL*) is preserved?
- And in what sense?
  - "strong" preservation: positive and negative results carry over
  - "weak" preservation: only positive results carry over
  - "match": logic equivalence coincides with formal relation

Checking trace equivalence is PSPACE-complete