Automatic verification of probabilistic systems

Jeremy Sproston
(docente di LFT laboratorio)
DipInfo si racconta
5 February 2019
Automatic verification

• **Aim:**
  – Reliable and efficient systems
  – Detection of errors and inefficiencies early in system development

• **Verification of system designs:**
  – Establish absence of errors and inefficiencies in the system design

• **Automatic verification:**
  – Limit burden on system developer to formally modelling the system (+ requirements)
  – Checking the system model is done by a computer (push-button)

• **Model checking:**
  – Example of an **exhaustive** automatic verification method
  – 2007 Turing award won by Clarke, Emerson and Sifakis for initiating and developing model checking
Model checking: automatic verification of probabilistic systems

Jeremy Sproston
(docente di LFT laboratorio)
DipInfo si racconta
5 February 2019
Model checking

System → System model → Model checking tool

System satisfies requirement(s)

System does not satisfy requirement(s)

∃ ⅀ error
∀ □ (request → ◇ response)
Probabilistic model checking

- Not all behaviours of systems are equally likely
  - E.g., randomised algorithms, physical faults, human behaviour, meteorological conditions
- A system design may be considered as acceptable if undesirable behaviour is exhibited with sufficiently low probability
- Model the system (and requirement) with probabilities
Probabilistic model checking

System satisfies requirement(s) (Boolean result)

System satisfies requirement(s) with certain probability (quantitative result)

System does not satisfy requirement(s) (Boolean result)

System

System model

Model checking tool

Temporal logic formulas

System requirements

System model

Model checking tool

Temporal logic formulas

System requirements

System satisfies requirement(s) (Boolean result)

System satisfies requirement(s) with certain probability (quantitative result)

System does not satisfy requirement(s) (Boolean result)

\[ P_{\leq 0.05}[^{\exists} \text{error}] \]
\[ P_{=1}[^{\Box} \text{(request } \rightarrow P_{\geq 0.95}[^{\Diamond} \text{response}]})] \]
\[ P_{\geq 0.99}[^{\Diamond} \text{success}] \]
Applications of probabilistic model checking

- Case studies considered with the probabilistic model checking tools PRISM or STORM
  - Randomised protocols
    (e.g., IEEE 802.3 CSMA/CD protocol, IEEE 802.11 wireless LANs, IEEE 802.15.4 CSMA-CA protocol, IPv4 Zeroconf protocol, IEEE 1394 FireWire root contention protocol, Bluetooth device discovery, bounded retransmission protocol, …)
  - Planning problems
    (e.g., human-in-the-loop UAV mission planning, planning for autonomous systems sharing an environment with humans, …)
  - Biological systems
    (e.g., Fibroblast Growth Factor signaling, DNA walkers, …)
  - Performance, reliability, security, randomised distributed algorithms, …
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

Art by Russ Nicholson
https://russnicholson.blogspot.com/
The player rolls a pair of six sided dice and adds the total to the hero's SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

**HERO**
- SKILL=8
- STAMINA=6

**OPPONENT**
- SKILL=7
- STAMINA=4
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

HERO
SKILL=8
STAMINA=6

OPPONENT
SKILL=7
STAMINA=4
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

**Hero**
- Skill: 8
- Stamina: 4

**Opponent**
- Skill: 7
- Stamina: 4

8+6+1=15 ≤ 16=7+3+6

Hero loses combat round
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero's SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

<table>
<thead>
<tr>
<th>Hero</th>
<th>Opponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKILL=8</td>
<td>SKILL=7</td>
</tr>
<tr>
<td>STAMINA=4</td>
<td>STAMINA=4</td>
</tr>
</tbody>
</table>
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

<table>
<thead>
<tr>
<th>HERO</th>
<th>OPPONENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKILL=8</td>
<td>SKILL=7</td>
</tr>
<tr>
<td>STAMINA=4</td>
<td>STAMINA=4</td>
</tr>
</tbody>
</table>
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

HERO
SKILL=8  
STAMINA=4

OPPONENT
SKILL=7  
STAMINA=4

8+3+2=13 ≥ 10=7+2+1
HERO wins combat round
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

 HERO
skill=8
STAMINA=4

OPPONENT
skill=7
STAMINA=2

8+3+2=13 ≥ 10=7+2+1
HERO wins combat round
The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

**HERO**
- SKILL = 8
- STAMINA = 4

**OPPONENT**
- SKILL = 7
- STAMINA = 2
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

**HERO**
SKILL=8
STAMINA=4

**OPPONENT**
SKILL=7
STAMINA=2
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

HERO
SKILL=8
STAMINA=4

OPPONENT
SKILL=7
STAMINA=2

8+6+3=17 ≥ 13=7+3+3
HERO wins combat round
Basic rules for combat

The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. This process continues until one party's STAMINA reaches 0, at which point they are dead.

<table>
<thead>
<tr>
<th>Hero</th>
<th>Opponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKILL=8</td>
<td>SKILL=7</td>
</tr>
<tr>
<td>STAMINA=4</td>
<td>STAMINA=0</td>
</tr>
</tbody>
</table>

8+6+3=17 ≥ 13=7+3+3

HERO wins combat round
(Non-probabilistic) model: graph

Single round

Hero STAMINA=6, opponent STAMINA=4
(Non-probabilistic) model checking

- Example of requirement: the hero can win the battle
- More formally: there exists a path that visits a state for which the opponent’s STAMINA equals 0
- Temporal logic formula: $\exists \diamond (s_o = 0)$
- Model checking (symbolic): reach backwards from states with $s_o = 0$

Hero STAMINA=6, opponent STAMINA=4
(Non-probabilistic) model checking

- Example of requirement: the hero can win the battle
- More formally: there exists a path that visits a state for which the opponent’s STAMINA equals 0
- Temporal logic formula: $\exists\diamond (s_o = 0)$
- Model checking (symbolic): reach backwards from states with $s_o = 0$
(Non-probabilistic) model checking

- Example of requirement: the hero can win the battle
- More formally: there exists a path that visits a state for which the opponent’s STAMINA equals 0
- Temporal logic formula: $\exists \Diamond (s_o = 0)$
- Model checking (symbolic): reach backwards from states with $s_o = 0$
(Non-probabilistic) model checking

- Example of requirement: the hero can win the battle
- More formally: there exists a path that visits a state for which the opponent’s STAMINA equals 0
- Temporal logic formula: $\exists \diamond (s_o = 0)$
- Model checking (symbolic): reach backwards from states with $s_o = 0$

States reaching states with $s_o = 0$ in two steps; cannot add further states

Hero STAMINA=6, opponent STAMINA=4

States reaching states with $s_o = 0$ in two steps; cannot add further states
(Non-probabilistic) model checking

- Example of requirement: the hero can win the battle
- More formally: there exists a path that visits a state for which the opponent’s STAMINA equals 0
- Temporal logic formula: \( \exists \Diamond (s_o = 0) \)
- Model checking (symbolic): reach backwards from states with \( s_o = 0 \)

Hero STAMINA=6, opponent STAMINA=4

Cannot add further states: computed set of states satisfying \( \exists \Diamond (s_o = 0) \)
Model: Markov chain

Single round

Probability of a draw

Probability opponent wins

Probability hero wins

Hero SKILL=8, STAMINA=6,

opponent SKILL=7, STAMINA=4

145/432

35/324

721/1296

1

1

1

1

1
Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8

More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8

Temporal logic formula: $P_{\geq 0.8}[\Diamond (s_o=0)]$

Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o=0$ and probability 0 to all other states
Model: Markov chain

- Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8
- More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8
- Temporal logic formula: $P_{\geq 0.8}[\Diamond (s_o=0)]$
- Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o=0$ and probability 0 to all other states

Hero SKILL=8, STAMINA=6,
opponent SKILL=7, STAMINA=4

0 steps
Model: Markov chain

- Example of (qualitative) requirement: **the hero wins the battle with probability at least 0.8**
- More formally: **the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8**
- Temporal logic formula: \( P \geq 0.8[\Diamond(s_o=0)] \)
- Model checking (value iteration): **propagate probabilities backwards, starting by assigning probability 1 to states with \( s_o = 0 \) and probability 0 to all other states**

Hero SKILL=8, STAMINA=6, opponent SKILL=7, STAMINA=4

1 step
Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8

More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8

Temporal logic formula: $P_{\geq 0.8}[\Diamond (s_o = 0)]$

Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o = 0$ and probability 0 to all other states

2 steps
Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8

More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8

Temporal logic formula: $P_{\geq 0.8}[\Diamond(s_o=0)]$

Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o=0$ and probability 0 to all other states
Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8

More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8

Temporal logic formula: $P \geq 0.8[\Diamond (s_o = 0)]$

Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o = 0$ and probability 0 to all other states

4 steps
• Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8
• More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8
• Temporal logic formula: $P \geq 0.8[\diamond (s_o=0)]$
• Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o=0$ and probability 0 to all other states
Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8

More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8

Temporal logic formula: $P \geq 0.8[\Diamond(s_o=0)]$

Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o=0$ and probability 0 to all other states

6 steps
Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8
More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8
Temporal logic formula: $P_{\geq 0.8}[\Diamond (s_o=0)]$
Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o=0$ and probability 0 to all other states
Model: Markov chain

- Example of (qualitative) requirement: the hero wins the battle with probability at least 0.8
- More formally: the probability of paths that visit a state for which the opponent’s STAMINA equals 0 is at least 0.8
- Temporal logic formula: $\mathbb{P}_{\geq 0.8}[\Diamond (s_o = 0)]$
- Model checking (value iteration): propagate probabilities backwards, starting by assigning probability 1 to states with $s_o = 0$ and probability 0 to all other states
Results for multiple Markov chains

What is the probability that the hero (with different levels of SKILL and STAMINA) wins, against different opponents? (Results obtained by PRISM)
The player rolls a pair of six sided dice and adds the total to the hero’s SKILL, then does the same for their opponent. Whichever combatant has scored higher has wounded the other, and the wounded party must subtract 2 points from their STAMINA. **At this point the player has the option to Test Your Luck, a gamble which either increases or decreases the damage done. This process continues until one party's STAMINA reaches 0, at which point they are dead.**
Rules with LUCK

Test Your Luck: the player rolls a pair of six sided dice and compares the result to their LUCK score. If the result is lower than or equal to their LUCK score, the LUCK roll is successful, otherwise it is unsuccessful. The player's LUCK score is decreased by 1 each time it is tested and thus subsequent Tests of Luck become increasingly difficult.

Testing Your Luck when the wounded party is the opponent:
If Test Your Luck is successful, the opponent loses 4 STAMINA
If Test Your Luck is unsuccessful, the opponent loses only 1 STAMINA
Rules with LUCK

*Test Your Luck*: the player rolls a pair of six sided dice and compares the result to their LUCK score. If the result is lower than or equal to their LUCK score, the LUCK roll is *successful*, otherwise it is *unsuccessful*. The player's LUCK score is decreased by 1 each time it is tested and thus subsequent *Tests of Luck* become increasingly difficult.

*Testing Your Luck* when the wounded party is the hero:
If *Test Your Luck* is *successful*, the hero loses only 1 STAMINA
If *Test Your Luck* is *unsuccessful*, the hero loses 3 STAMINA
Model: Markov decision process

Single round

- **s_p s_o L**: Probability of a draw
- **s_p s_o L**: Probability of the hero winning
- **s_p s_o L**: Probability of the opponent winning

States:
- **s_p s_o L**: Hero state
- **s_p s_o L**: Opponent state
- **s_p s_o L**: Success state
- **s_p s_o L**: Failure state

Actions:
- Choose not to test luck
- Choose to test luck

Transitions:
- Probability of successful LUCK roll
- Probability of unsuccessful LUCK roll

Choices:
- Choose not to test luck
- Choose to test luck
Results for Markov decision processes

- Using LUCK in the most favourable (maximum probability)/unfavourable (minimum probability) way, what is the probability that the hero wins?
Probabilistic model checking: overview

• Current tools: typically able to handle models with around $10^6$ states in around 3 minutes on a standard PC
• Applications to the verification of protocols, planning, security, systems biology, etc.
• Extended in multiple directions: multiple objectives, statistical model checking (link with simulation), abstraction/approximation (link with learning), compositional reasoning, games, strategy representation, etc.

• Other topic of my interest: model checking of timed systems
  – System behaviours may have duration
  – Timed automata: widespread diffusion in the model-checking field
  – My work: probabilistic timed automata, a combination of Markov decision processes and timed automata