Probabilistic Model Checking

Dave Parker

University of Oxford

AVoCS '07
Overview

• Introduction and motivation
• Probabilistic models
  – discrete-time Markov chains (DTMCs)
  – Markov decision processes (MDPs)
• Property specification
  – the logic PCTL, quantitative specifications, costs and rewards
• Probabilistic model checking
  – algorithms, implementation, tool support
• Case studies
  – probabilistic contract signing
Verification via model checking

- EF error
- Finite-state model
- Temporal logic specification
- Model checker
- Error trace
- Result

AVOCS'07
Probabilistic model checking

Finite-state probabilistic model
e.g. Markov chain

Probabilistic model checker
e.g. PRISM

Result

Probabilistic temporal logic specification
e.g. PCTL

Quantitative results

$P_{<0.01}$ [ F error ]

AVOCS'07
Why probability?

- **Randomisation used in distributed coordination algorithms**
  - as a symmetry breaker, e.g. Bluetooth, FireWire, ...
  - in gossip routing to reduce flooding
- **To model uncertainty and performance**
  - failure rates, e.g. communication protocols
  - component life-times, e.g. network cluster
  - inter-arrival times, e.g. queuing systems
  - biochemical reaction rates
- **Requirements are quantitative as well as qualitative**
  - Quality of Service: how reliable is my car’s Bluetooth network?
  - how efficient is my phone’s power management policy?
  - quantification of trust, anonymity, ...
Overview

- Introduction and motivation
- **Probabilistic models**
  - discrete-time Markov chains (DTMCs)
  - Markov decision processes (MDPs)
- Property specification
  - the logic PCTL, quantitative specifications, costs and rewards
- Probabilistic model checking
  - algorithms, implementation, tool support
- Case studies
  - probabilistic contract signing
Discrete–time Markov chains (DTMCs)

• State–transition systems augmented with probabilities
  – discrete set of states representing possible system configurations
  – transitions between states occur in discrete time–steps
  – probability of making transitions between states is given by discrete probability distributions

• Formally, a DTMC D is a tuple \((S,s_{\text{init}},P,L)\)
  – \(S\) is a finite set of states, \(s_{\text{init}} \in S\) is the initial state
  – \(P : S \times S \to [0,1]\) is the transition probability matrix
    where \(\Sigma_{s' \in S} P(s,s') = 1\) for all \(s \in S\)
  – \(L : S \to 2^{AP}\) labels states with atomic propositions
Paths and probabilities

• A (finite or infinite) path through a DTMC
  – is a sequence of states \( s_0s_1s_2s_3\ldots \) such that \( P(s_i,s_{i+1}) > 0 \ \forall i \)
  – represents an execution (i.e. one possible behaviour) of the system which the DTMC is modelling

• To reason (quantitatively) about this system
  – need to define a probability space over paths [KSK76]

• Intuitively:
  – sample space: \( \text{Path}(s) = \text{set of all infinite paths from state } s \)
  – events: sets of infinite paths from \( s \)
  – basic events: cylinder sets (or “cones”)
    – cylinder set \( C(\omega) \), for a finite path \( \omega \)
      = set of infinite paths with the common finite prefix \( \omega \)
  – probability measure \( Pr_s \) over \( \text{Path}(s) \)
  – for example: \( Pr_s(C(ss_1s_2) = P(s,s_1)\cdot P(s_1,s_2) \)
Adding nondeterminism

• But, some aspects of a system may not be probabilistic and should not be modelled probabilistically; for example:

  • Concurrency – scheduling of parallel components
    – e.g. randomised distributed algorithms – multiple probabilistic processes operating asynchronously

  • Unknown environments
    – e.g. probabilistic security protocols – unknown adversary

  • Underspecification – unknown model parameters
    – e.g. a probabilistic communication protocol designed for message propagation delays of between $d_{\text{min}}$ and $d_{\text{max}}$
Markov decision processes (MDPs)

- Extension of DTMCs which allow nondeterministic choice
  - discrete set of states
  - transitions are discrete time-steps
  - nondeterministic choice between several discrete probability distributions over successor states

- Formally, an MDP M is a tuple \((S, s_{\text{init}}, \text{Steps}, L)\) where:
  - States \(S\), initial state \(s_{\text{init}}\) and labelling function \(L\) as for DTMCs
  - \(\text{Steps} : S \rightarrow 2^{\text{Act} \times \text{Dist}(S)}\) is the transition probability function
    where \(\text{Act}\) is a set of actions and \(\text{Dist}(S)\) is the set of discrete probability distributions over the set of states \(S\)
Paths, probabilities, adversaries

- A (finite or infinite) path through an MDP
  - is a sequence of states and action/distribution pairs
  - $s_0(a_0, \mu_0)s_1(a_1, \mu_1)s_2\ldots$ where $(a_i, \mu_i) \in \text{Steps}(s_i)$ and $\mu_i(s_{i+1}) > 0$
  - note both nondeterministic and probabilistic choice resolved

- To reason about probabilistic behaviour in an MDP
  - first resolve nondeterminism with an adversary
  - adversaries are also known as “schedulers” or “policies”
  - formally, an adversary is a mapping from any finite path in the MDP to a subsequent action/distribution pair

- For an adversary $A$
  - the MDP reduces to a (possibly infinite state) DTMC
  - the resulting set of infinite paths is $\text{Path}^A(s)$
  - over which we can define a probability measure $\text{Pr}^A_s$
Other models

• **Continuous–time Markov chains (CTMCs)**
  – discrete state space, continuous time
  – exponentially distributed delays
  – suited to modelling component lifetimes, inter–arrival times, biochemical reaction rates, …

• **Probabilistic timed automata (PTAs)**
  – probabilistic extension of timed automata
  – discrete states, real–time clocks, discrete probability distributions, nondeterminism

• **Others:** continuous–time MDPs, interactive Markov chains, …
Overview

• Introduction and motivation
• Probabilistic models
  – discrete-time Markov chains (DTMCs)
  – Markov decision processes (MDPs)
• Property specification
  – the logic PCTL, quantitative specifications, costs and rewards
• Probabilistic model checking
  – algorithms, implementation, tool support
• Case studies
  – probabilistic contract signing
PCTL

- **Temporal logic for describing properties of DTMCs/MDPs**
  - PCTL = Probabilistic Computation Tree Logic [HJ94]
  - essentially the same as the logic pCTL of [ASB+95]

- **Extension of (non–probabilistic) temporal logic CTL**
  - key addition is probabilistic operator P
  - quantitative extension of CTL’s A and E operators

- **Example**
  - send → P_{≥0.95} [ F_{≤10} deliver ]
  - “if a message is sent, then the probability of it being delivered within 10 steps is at least 0.95”
PCTL syntax

- \( \phi ::= \text{true} \mid a \mid \phi \land \phi \mid \neg \phi \mid P_{\sim p} [ \psi ] \)  
  
  (state formulas)

- \( \psi ::= X \phi \mid \phi U \leq k \phi \mid \phi U \phi \)  
  
  (path formulas)

- where \( a \) is an atomic proposition, used to identify states of interest, \( p \in [0,1] \) is a probability, \( \sim \in \{<,>,\leq,\geq\}, k \in \mathbb{N} \)

- Usual equivalences:
  - \( \text{false} \equiv \neg \text{true}, \phi_1 \lor \phi_2 \equiv \neg (\neg \phi_1 \land \neg \phi_2), \phi_1 \rightarrow \phi_2 \equiv \neg \phi_1 \lor \phi_2 \)
  - Reachability (“eventually”): \( F \phi \equiv \text{true} U \phi, \ F \leq k \phi \equiv \text{true} U \leq k \phi \)
  - Invariance (“always”): \( G \phi \equiv \neg (F \neg \phi) \equiv \neg (\text{true} U \neg \phi) \)
PCTL semantics for DTMCs

- **PCTL formulas interpreted over states of a DTMC**
  - usual (CTL) semantics for most operators

- **Semantics of the probabilistic operator P**
  - informal definition: \( s \models P_{\sim p} [ \psi ] \) means that “the probability, from state \( s \), that \( \psi \) is true for an outgoing path satisfies \( \sim p \)”
  - example: \( s \models P_{<0.25} [ X \text{ fail} ] \iff \text{“the probability of atomic proposition fail being true in the next state of outgoing paths from } s \text{ is less than 0.25”} \)
  - formally: \( s \models P_{\sim p} [ \psi ] \iff \Pr_s \{ \omega \in \text{Path}(s) \mid \omega \models \psi \} \sim p \)
PCTL semantics for MDPs

- **Semantics of the probabilistic operator P**
  - can only define probabilities for a specific adversary A
  - $s \models P_{\neg p} [\psi]$ means “the probability, from state $s$, that $\psi$ is true for an outgoing path satisfies $\neg p$ for all adversaries”
  - formally $s \models P_{\neg p} [\psi] \iff \text{Prob}^A(s, \psi) \sim p$ for all adversaries $A$
  - where $\text{Prob}^A(s, \psi) = \Pr^A_s \{ \omega \in \text{Path}^A(s) | \omega \models \psi \}$

- **In practice, we compute:**
  - $p_{\text{max}}(s, \psi) = \sup_A \text{Prob}^A(s, \psi)$
  - $p_{\text{min}}(s, \psi) = \inf_A \text{Prob}^A(s, \psi)$

- **Can also define semantics for a specific class of adversaries**
  - e.g. for all fair adversaries
Quantitative properties

- Consider a PCTL formula $P_{\sim p} [ \psi ]$
  - if the probability is unknown, how to choose the bound $p$?
- PRISM allows (for DTMCs) properties of the form $P=? [ \psi ]$
  - when the outermost operator of a PTCL formula is $P$
  - “what is the probability that path formula $\psi$ is true?”
  - model checking is no harder: compute the values anyway
- And for MDPs:
  - properties of the form $P_{\min=?} [ \psi ]$ and $P_{\max=?} [ \psi ]$
- Experiments: ranges of model/property parameters
  - e.g. $P=? [ F^{\leq T} \text{ error} ]$ for $T=1..100$, $N=1..5$
    - where $T$ is a time bound and $N$ is some model parameter
  - identify patterns, trends, anomalies in quantitative results
Some real PCTL examples

• NAND multiplexing system
  – $P_\omega [ F \text{ err/total} > 0.1 ]$
  – “what is the probability that 10% of the NAND gate outputs are erroneous?”

• FireWire communication protocol
  – $P_\omega [ F_{\leq t} \text{ done}_1 \lor \text{ done}_2 ]$
  – “what is the probability that a leader node has been elected within $t$ clock-ticks?”

• Security: EGL contract signing protocol
  – $P_\omega [ F (\text{pairs}_a = 0 \land \text{pairs}_b > 0) ]$
  – “what is the probability that the party B gains an unfair advantage during the execution of the protocol?”
Probability that 10% of gate outputs are erroneous for varying gate failure rates and numbers of stages

Optimum probability of leader election by time T for various coin biases

Probability that parties gain unfair advantage for varying numbers of secret packets sent
Best/worst-case scenarios

- Combining “quantitative” and “exhaustive” aspects

- All possible resolutions of nondeterminism (MDPs)
  - $P_{\min} = \min \{ I \} \cup \{ F \} \leq t$ - “minimum probability of process 1 finishing before process 2, for any scheduling of processes?”

- Computing values for a range of states
  - $P = \min \{ I \} \cup \{ F \} \leq t$ - “minimum probability of the leader election algorithm completing within $t$ steps from any state where there are at most $k$ tokens”
Costs and rewards

• Augment DTMCs/MDPs with rewards (or, conversely, costs)
  – real-valued quantities assigned to states and/or transitions
  – no distinction between rewards (“good”) and costs (“bad”)
  – these can have a wide range of possible interpretations

• Some examples:
  – elapsed time, power consumption, size of message queue,
    number of messages successfully delivered, net profit, ...

• DTMC/MDP can have (multiple) reward structures \((\rho, \iota)\)
  – state reward function: \(\rho : S \to \mathbb{R}_{\geq 0}\)
  – transition reward function: \(\iota : S \times S \to \mathbb{R}_{\geq 0}\) (for DTMCs)
    \(\iota : S \times \text{Act} \to \mathbb{R}_{\geq 0}\) (for MDPs)
Cost- and reward-based properties

• Formal specification of properties relating to rewards
  – add a new R operator to PCTL, similar to existing P operator
  – reason about expected values of rewards
  – two different interpretations: instantaneous and cumulative

• Instantaneous reward properties
  – state rewards only; state-based measures: “queue size”, “number of open channels”, “concentration of reactant X”, ...
  – \( R = ? \ [ I = t ] \) e.g. “expected message queue size at time t?”

• Cumulative reward properties
  – state and transition rewards, e.g. “time”, “power consumption”, “number of messages lost”
  – \( R = ? \ [ F \ end ] \) e.g. “expected time for protocol termination?”
  – \( R_{\max} = ? \ [ C \leq 2 ] \) e.g. “maximum expected power consumption during the first 2 hours that the system is in operation?”
Maximum expected time for leader election for various coin biases

Worst-case expected number of steps to stabilise for initial configurations with K tokens amongst N processes

Expected reactant concentrations over the first 12 hours
Overview

• Introduction and motivation
• Probabilistic models
  – discrete-time Markov chains (DTMCs)
  – Markov decision processes (MDPs)
• Property specification
  – the logic PCTL, quantitative specifications, costs and rewards
• Probabilistic model checking
  – algorithms, implementation, tool support
• Case studies
  – probabilistic contract signing
Probabilistic model checking

- **Two distinct phases:**
  - **model construction**, i.e. translation from high-level modelling language (PRISM language, process algebra, ...) to DTMC/MDP
  - **model checking** – qualitative or quantitative PCTL properties

- **Probabilistic model checking algorithms**
  - **graph-based algorithms** on underlying transition system
    - reachability, qualitative probabilistic reachability
  - **numerical computation** – calculation of probabilities, rewards
    - usually, linear equation systems or linear optimisation problems
    - typically use iterative methods, e.g. Gauss–Seidel, value iteration
  - also: **simulation-based sampling** for approximate analysis

- **State-space explosion problem**
  - considerable effort devoted to **efficient** implementations
  - PRISM uses **symbolic** (BDD-based) data structures, techniques
Tool support

• **Probabilistic model checkers**
  - **PRISM**: DTMCs, MDPs, CTMCs + rewards
  - **ETMCC/MRMC**: DTMCs, CTMCs + reward extensions
  - **LiQuor**: LTL verification for MDPs (Probmela language)
  - **RAPTURE**: prototype for abstraction/refinement of MDPs
  - Simulation–based tools: APMC, Ymer, VESTA
  - CSL model checking for CTMCs: APNN–Toolbox, SMART
  - Multiple formalism/tool solutions: CADP, Möbius

• **Tool support provides:**
  - high–level languages/formalisms for building models
  - fully automated verification, efficient implementations
  - automation, visualisation of quantitative results
  - discrete event simulations– debugging, approximations
Overview

• Introduction and motivation
• Probabilistic models
  – discrete-time Markov chains (DTMCs)
  – Markov decision processes (MDPs)
• Property specification
  – the logic PCTL, quantitative specifications, costs and rewards
• Probabilistic model checking
  – algorithms, implementation, tool support
• Case studies
  – probabilistic contract signing
PRISM case studies

- **Communication and multimedia protocols**
  - Bluetooth device discovery, FireWire root contention, IPv4 Zeroconf protocol, IEEE 802.3 CSMA/CD protocol, IEEE 802.11 WiFi wireless LANs, Zigbee (IEEE 802.15.4)

- **Security systems/protocols**
  - Probabilistic contract signing, Crowds protocol (anonymity), Probabilistic fair exchange, PIN cracking schemes, Negotiation frameworks, Quantum cryptography

- **Randomised distributed algorithms for:**
  - Byzantine agreement, Consensus, Self-stabilisation, Leader election, Mutual exclusion, ...

- **Analysis of behaviour/performance/reliability of:**
  - Biochemical reaction networks, Dynamic power management, Dynamic voltage scaling algorithms, Manufacturing/control systems, NAND multiplexing, Groupware protocols, ...
Case Study – Contract signing

- **Two parties want to agree on a contract**
  - each will sign if the other will sign, but do not trust each other
  - in real life, sit down and write signatures simultaneously
  - on an asynchronous network such as the internet?
  - no deterministic protocol without use of a trusted third party

- **Partial secret exchange protocol – randomised solution**
  - Even, Goldreich, Lempel [EGL85]
  - “viability” – with correct execution of protocol, signatures successfully exchanged
  - “approximate concurrency” – if one party X executes protocol correctly, with very high probability can always obtain signature of Y with approximately same work as Y
Contract signing – EGL protocol

- Partial secret exchange protocol for 2 parties (A and B)

- A (B) holds 2N secrets \(a_1, \ldots, a_{2N} \ (b_1, \ldots, b_{2N})\)
  - a secret is a binary string of length \(L\)
  - secrets partitioned into pairs: e.g. \(\{ (a_i, a_{N+i}) \mid i=1, \ldots, N \}\)
  - A (B) committed if B (A) knows one of A’s (B’s) pairs

- Uses “1–out–of–2 oblivious transfer protocol” \(OT(S,R,x,y)\)
  - S sends \(x\) and \(y\) to R
  - R receives \(x\) with probability \(\frac{1}{2}\) otherwise receives \(y\)
  - S does not know which one R receives
  - if S cheats then R can detect this with probability \(\frac{1}{2}\)
Contract signing – EGL protocol

(step 1)
for ( i=1,…,N )
    OT( A, B, a_i, a_{N+i} )
    OT( B, A, b_i, b_{N+i} )

(step 2)
for ( i=1,…,L ) (where L is the bit length of the secrets)
    for ( j=1,…,2N )
        A transmits bit i of secret a_j to B
    for ( j=1,…,2N )
        B transmits bit i of secret b_j to A
EGL protocol – Step 1

(repeat for $i = 1 \ldots N$)
EGL protocol – Step 2

Party A

A sends bit $i$ of $a_j$ to B for $j=1\ldots2N$

Then B does the same for $b_j$

(repeat for $i=1\ldots L$)

Party B
Contract signing – Results

- Modelled in PRISM as a DTMC (no concurrency) [NS06]

- Discovered a weakness in the protocol
  - party B can act maliciously by quitting the protocol early
  - this behaviour not considered in the original analysis

- PRISM analysis shows
  - if B stops participating in the protocol as soon as he/she has obtained one of A’s pairs, then, with probability 1:
    - B will end up possessing a pair of A’s secrets
    - A will not have complete knowledge of any pair of B’s secrets
  - protocol is not fair under this attack:
    - B has a distinct advantage over A
Contract signing – Results

• The protocol is unfair because in step 2:
  – A sends a bit for each of its secret before B does

• Can we make this protocol fair by changing the message sequence scheme?

• Since the protocol is asynchronous, the best we can hope for is:
  – B (or A) has this advantage with probability $\frac{1}{2}$

• We consider 3 possible alternative message sequence schemes...
Contract signing – EGL2

(step 1)
...
(step 2)
for ( i=1,…,L )
  for ( j=1,…,N ) A transmits bit i of secret $a_j$ to B
  for ( j=1,…,N ) B transmits bit i of secret $b_j$ to A
for ( j=N+1,…,2N ) A transmits bit i of secret $a_j$ to B
for ( j=N+1,…,2N ) B transmits bit i of secret $b_j$ to A
Modified step 2 for EGL2

A sends bit $i$ of $a_j$ to B for $j=1\ldots N$

Then B does the same for $b_j$

(after $j=1\ldots N$, send $j=N+1\ldots 2N$)

(then repeat for $i=1\ldots L$)
(step 1) … 

(step 2) 
for ( i=1,...,L ) for ( j=1,...,N )  
A transmits bit i of secret $a_j$ to B  
B transmits bit i of secret $b_j$ to A  
for ( i=1,...,L ) for ( j=N+1,...,2N )  
A transmits bit i of secret $a_j$ to B  
B transmits bit i of secret $b_j$ to A
Modified step 2 for EGL3

Party A

1...L

1...N

N+1...2N

Party B

1...L

1...N

N+1...2N

A sends bit $i$ of $a_j$ to B for

Then B does the same for $b_j$

(repeat for $j=1...N$ and for $i=1...L$)

(then send $j=N+1...2N$ for $i=1...L$)
Contract signing – EGL4

(step 1)
...
(step 2)
for (i=1,...,L)
  A transmits bit i of secret $a_1$ to B
  for (j=1,...,N) B transmits bit i of secret $b_j$ to A
  for (j=2,...,N) A transmits bit i of secret $a_j$ to B
for (i=1,...,L)
  A transmits bit i of secret $a_{N+1}$ to B
  for (j=N+1,...,2N) B transmits bit i of secret $b_j$ to A
  for (j=N+2,...,2N) A transmits bit i of secret $a_j$ to B
Modified step 2 for EGL4

Party A
1...L
1...N
N+1...2N
A sends bit $i$ of $a_1$ to B

Then B sends bit $i$ of $b_j$ to B for $j=1...N$

Then A sends bit $i$ of $a_j$ to B for $j=2...N$

(repeat for $i=1...L$)
(then send $j=N+1...2N$ in same fashion)

Party B
1...L
1...N
N+1...2N
Contract signing – Results

• The chance that the protocol is unfair
  – probability that one party gains knowledge first
  – $P = \Pr [\neg \text{know}_A \land \text{know}_B]$ and $P = \Pr [\neg \text{know}_B \land \text{know}_A]$
Contract signing – Results

• How unfair the protocol is to each party
  – expected number of bits that a party needs to know a pair once the other party knows a pair
  – need to modify the model and define a reward structure
  – dependent on which party we are considering

• Expected number of bits that A needs to know a pair once B knows a pair
  – add a transition to a new state labelled by “done” as soon as B knows a pair
  – assign a reward equal to the number of bits that A requires to know a pair to this transition
  – check the formula $R \geq \mathbb{E}[F \text{ done}]$
Contract signing – Results

- How unfair the protocol is to each party
  - expected number of bits that a party needs to know a pair once the other party knows a pair
Contract signing – Results

• The influence that each party has on the fairness
   – once a party knows a pair, the expected number of messages from this party required before the other party knows a pair
   – measures the influence as a corrupted party can delay its messages
   – need to define a reward structure
   – dependent on which party we are considering

• Once B knows a pair, the expected number of messages from B required before A knows a pair
   – assign reward of 1 to transitions which correspond to B sending a message to A from a state where B knows a pair
   – check the formula $R_{\geq?}[F \text{ know}_A]$
Contract signing – Results

- The influence that each party has on the fairness
  - once a party knows a pair, the expected number of messages from this party required before the other party knows a pair
Contract signing – Results

• **The duration of unfairness of the protocol**
  – once a party knows a pair, the expected total number of messages that need to be sent (by either party) before the other knows a pair
  – need to define a reward structure
  – dependent on which party we are considering

• **Once B knows a pair, the expected total number of messages that need to be sent before A knows a pair**
  – assign reward of 1 to transitions which correspond to either party sending a message from a state where B knows a pair
  – check the formula $R = ?[F \text{ know}_A]$
Contract signing – Results

- The duration of unfairness of the protocol
  - once a party knows a pair, the expected total number of messages that need to be sent before the other knows a pair
Contract signing – Results

• Results show EGL4 is the ‘fairest’ protocol

• Except for duration of fairness measure…

• Expected messages that need to be sent for a party to know a pair once the other party knows a pair
  – this value is larger for B than for A
  – in fact, as N increases, this measure increases for B and decreases for A

• Solution
  – if a party sends a sequence of bits in a row (without the other party sending messages in between), require that the party send these bits as a single message
Contract signing – Results

- The duration of unfairness of the protocol
  - once a party knows a pair, the expected total number of messages that need to be sent before the other knows a pair
Summary

- **Summary**
  - probabilistic model checking
  - probabilistic models: DTMCs, MDPs, …
  - property specifications, implementation and tool support
  - case studies, probabilistic contract signing

- **For more information, see the PRISM web page**
  - [www.prismmodelchecker.org](http://www.prismmodelchecker.org)
  - 11 part lecture course
  - related publications, talks, tutorials, links
  - on-line example repository (40+ case studies)
  - PRISM tool download: binaries, source code (GPL)
  - on-line manual, tutorial