Task Scheduling and Execution for Long-Term Autonomy

Nick Hawes    Dave Parker

University of Birmingham


ICAPS Summer School, June 2016
Overview

• **Formal verification**
  – probabilistic model checking

• **Markov decision processes (MDPs)**
  – verification vs. strategy synthesis

• **Linear temporal logic (LTL)**
  – probabilistic model checking + MDPs + LTL

• **Multi-objective probabilistic model checking**
  – partially satisfiable task specifications
Formal verification

- **Formal verification**
  - the application of *rigorous*, mathematics–based techniques to check the *correctness* of computerised systems

- **Verifying probabilistic systems…**
  - *unreliable* or unpredictable behaviour
    - e.g. failures, message loss, delays, unreliable sensors/actuators
  - *randomisation* in algorithms/protocols
    - e.g. random back-off in communication protocols

- **We need to verify quantitative system properties**
  - “the probability of the airbag failing to deploy within 0.02 seconds of being triggered is at most 0.001”
  - not just correctness: reliability, timeliness, performance, …
  - not just verification: correctness by construction
Probabilistic model checking

• Construction and analysis of probabilistic models
  – state–transition systems labelled with probabilities (e.g. Markov chains, Markov decision processes)
  – from a description in a high–level modelling language

• Properties expressed in temporal logic, e.g. PCTL:
  – trigger → P_{\geq 0.999} [ F_{\leq 20} \text{ deploy } ]
  – “the probability of the airbag deploying within 20ms of being triggered is at least 0.999”
  – properties checked against models using exhaustive search and numerical computation
Probabilistic model checking

- **Key benefits**
  - *exact* results: guarantees, optimality, ...
  - fully automated, tools available (e.g. PRISM)
  - wide range of models, properties expressible

- **Key challenges**
  - scalability! state space explosion problem
  - results are only as good as the model

- **Application domains**
  - network/communication protocols, security, biology, power management, robotics & planning, ...
Markov decision processes (MDPs)

- Markov decision processes (MDPs)
  - model **nondeterministic** as well as **probabilistic** behaviour

![MDP Diagram]

- **Nondeterminism for:**
  - control: decisions made by a controller or scheduler
  - adversarial behaviour of the environment
  - concurrency/scheduling: interleavings of parallel components
  - abstraction, or under-specification, of unknown behaviour
• **A strategy** (or “policy”, “adversary”, “scheduler”)
  – is a resolution of nondeterminism, based on history
  – i.e. a mapping from finite paths to (distributions over) actions
  – induces (infinite–state) Markov chain (and probability space)

• **Classes of strategies:**
  – **memory**: memoryless, finite–memory, or infinite–memory
  – **randomisation**: deterministic or randomised
1. Verification
   - quantify over all possible strategies (i.e. best/worst-case)
   - $P_{\leq 0.1} [ F \text{ err }]$ : “the probability of an error occurring is $\leq 0.1$ for all strategies”
   - applications: randomised communication protocols, randomised distributed algorithms, security, ...

2. Controller synthesis
   - generation of "correct-by-construction" controllers
   - $P_{\leq 0.1} [ F \text{ err }]$ : "does there exist a strategy for which the probability of an error occurring is $\leq 0.1$?"
   - applications: robotics, power management, security, ...

Two dual problems; same underlying computation:
   - compute optimal (minimum or maximum) values
Running example

- Example MDP
  - robot moving through terrain divided into 3 x 2 grid
Larger example

Task scheduler  Map generator

Navigation planner  Motion planner
Example – Reachability

Verify: $P_{\leq 0.6} [F \text{goal}_1 ]$

or

Synthesise for: $P_{\geq 0.4} [F \text{goal}_1 ]$

⇓

Compute: $P_{\text{max}=?} [F \text{goal}_1 ]$

Optimal strategies: memoryless and deterministic

Computation:
graph analysis + numerical soln. (linear programming, value iteration, policy iteration)
Example – Reachability

Optimal strategy:

\[ s_0 : \text{east} \]
\[ s_1 : \text{south} \]
\[ s_2 : \_ \]
\[ s_3 : \_ \]
\[ s_4 : \text{east} \]
\[ s_5 : \_ \]

Verify: \( P \leq 0.6 \) \( [ F \ \text{goal}_1 ] \)

or

Synthesise for: \( P \geq 0.4 \) \( [ F \ \text{goal}_1 ] \)

⇓

Compute: \( P_{\text{max}} = 0.5 \) \( [ F \ \text{goal}_1 ] \)

Optimal strategies: memoryless and deterministic

Computation:

graph analysis + numerical soln.
(linear programming, value iteration, policy iteration)
Linear temporal logic (LTL)

- Logic for describing properties of executions [Pnueli]

- **LTL syntax:**
  - \( \psi ::= \text{true} \mid a \mid \psi \land \psi \mid \neg \psi \mid X \psi \mid \psi U \psi \mid F \psi \mid G \psi \)

- **Propositional logic + temporal operators:**
  - \( a \) is an atomic proposition (labelling a state)
  - \( X \psi \) means "\( \psi \) is true in the next state"
  - \( F \psi \) means “\( \psi \) is eventually true”
  - \( G \psi \) means “\( \psi \) remains true forever”
  - \( \psi_1 U \psi_2 \) means "\( \psi_2 \) is true eventually and \( \psi_1 \) is true until then”

- **Simple examples**
  - \( G \neg (\text{critical}_1 \land \text{critical}_2) \) – "the two processes never enter the critical section simultaneously"
  - \( \neg \text{error U end} \) – "the program terminates without any errors"
Linear temporal logic (LTL)

• LTL syntax:
  \[ \psi ::= \text{true} \mid a \mid \psi \land \psi \mid \neg\psi \mid X \psi \mid \psi \lor \psi \mid F \psi \mid G \psi \]

• Commonly used LTL formulae:
  - \( G (a \rightarrow F b) \) – "b always eventually follows a"
  - \( G (a \rightarrow X b) \) – "b always immediately follows a”
  - \( G F a \) – "a is true infinitely often"
  - \( F G a \) – "a becomes true and remains true forever"

• Robot task specifications in LTL
  - \( (G \neg \text{hazard}) \land (G F \text{ goal}_1) \) – "avoid hazard and visit goal\(_1\) infinitely often"
  - \( \neg \text{zone}_3 \lor (\text{zone}_1 \land (F \text{ zone}_4)) \) – "patrol zone 1 then 4, without passing through 3".
LTL for robot navigation

- **Probabilistic LTL on MDPs**
  - e.g. $P_{>0.7} [ (\neg \text{hazard}) \land (GF \text{ goal}_1) ]$ – "is the probability of avoiding hazard and visiting goal$_1$ infinitely often $> 0.7$?"
  - e.g. $P_{\max=?} [ \neg \text{zone}_3 \ U (\text{zone}_1 \land (F \text{ zone}_4)) ]$ – "max. probability of patrolling zones 1 then 4, without passing through 3?"

- **LTL + expected costs/times on MDPs**
  - minimise expected time to satisfy (co–safe) LTL formulas

- **Benefits of the approach**
  - LTL: flexible, unambiguous property specification
  - guarantees on performance ("correct by construction")
  - meaningful properties: probabilities, time, energy,…
    - c.f. ad–hoc reward structures, e.g. with discounting
  - efficient, fully–automated techniques
    - LTL–to–automaton conversion, MDP solution
Probabilistic model checking of LTL on MDPs

- convert LTL formula \( \psi \) to deterministic automaton \( A_\psi \) (Buchi, Rabin, finite, ...)
- build/solve product MDP \( M \otimes A_\psi \) (i.e. reduce to simpler problem)
- optimal strategies are deterministic, finite-memory

Det. Buchi automaton \( A_\psi \)
for \( \psi = G\neg h \land GF g_1 \)
Example: Product MDP construction

\[ M \otimes A_\psi \]

\[ \psi = G\neg h \land GF g \]
Example: Product MDP construction

$$M \otimes A_\psi$$
Co-safe LTL (and expected cost)

- Often focus on tasks completed in finite time
  - can restrict to co-safe fragment(s) of LTL
  - (any satisfying execution has a "good prefix")
  - e.g. $P_{\text{max}=?} \left[ \neg \text{zone}_3 \mathop{U} (\text{zone}_1 \land (F \text{ zone}_4)) \right]$
  - for simplicity, can restrict to syntactically co-safe LTL

- Expected cost/reward to satisfy (co-safe) LTL formula
  - e.g. $R_{\text{min}=?} \left[ \neg \text{zone}_3 \mathop{U} (\text{zone}_1 \land (F \text{ zone}_4)) \right]$ – "minimise exp. time to patrol zones 1 then 4, without passing through 3".

- Solution:
  - product of MDP and DFA
  - expected cost to reach accepting states in product
• Further use of probabilistic model checking…
  – (various probabilistic models, query languages)

• Nested queries
  – e.g. \( R_{\text{min}}=? \) [ safe U (zone\(_1\) \( \land \) (F zone\(_4\)) ) ] – "minimise exp. time to patrol zones 1 then 4, passing only through safe".
  – where safe denotes states satisfying \( \langle \langle \text{ctrl} \rangle \rangle R_{<2} [ F \text{ base } ] \) – "there is a strategy to return to base with expected time < 2"

• Analysis of generated controllers
  – expected power consumption to complete tasks?
  – conditional expectation, e.g. expected time to complete task, assuming it is completed successfully?
  – more detailed timing information (not just mean time)
Multi-objective model checking

- **Multi-objective probabilistic model checking**
  - investigate trade-offs between conflicting objectives
  - in PRISM, objectives are probabilistic LTL or expected costs

- **Achievability queries**: multi($P_{>0.95}[F\ send\ ],\ R_{time>10}\ [\ C\ ]$)
  - e.g. “is there a strategy such that the probability of message transmission is $> 0.95$ and expected battery life $> 10$ hrs?”

- **Numerical queries**: multi($P_{\text{max}}=?[F\ send\ ],\ R_{\text{time}>10}\ [\ C\ ]$)
  - e.g. “maximum probability of message transmission, assuming expected battery life-time is $> 10$ hrs?”

- **Pareto queries**:
  - multi($P_{\text{max}}=?[F\ send\ ],\ R_{\text{time max}}=?[\ C\ ]$)
  - e.g. "Pareto curve for maximising probability of transmission and expected battery life-time"
Multi-objective model checking

- Multi-objective probabilistic model checking
  - investigate trade-offs between conflicting objectives
  - in PRISM, objectives are probabilistic LTL or expected rewards
- Achievability queries: \( \text{multi}(P_{>0.95}[F \text{ send}], \text{R}_{\text{time}>10}[C]) \)
  - e.g. “is there a strategy such that the probability of message transmission is > 0.95 and expected battery life > 10 hrs?”
- Numerical queries: \( \text{multi}(P_{\text{max=?}}[F \text{ send}], \text{R}_{\text{time}>10}[C]) \)
  - e.g. “maximum probability of message transmission, assuming expected battery life–time is > 10 hrs?”
- Pareto queries:
  - \( \text{multi}(P_{\text{max=?}}[F \text{ send}], \text{R}_{\text{time max=?}}[C]) \)
  - e.g. "Pareto curve for maximising probability of transmission and expected battery life–time"
Multi-objective model checking

• **Optimal strategies:**
  – usually *finite-memory* (e.g. when using LTL formulae)
  – may also need to be *randomised*

• **Computation:**
  – construct a product MDP (with several automata), then reduces to linear programming [TACAS'07, TACAS'11]
  – can be approximated using iterative numerical methods, via approximation of the Pareto curve [ATVA'12]

• **Extensions** [ATVA'12]
  – arbitrary Boolean combinations of objectives
    • e.g. $\psi_1 \Rightarrow \psi_2$ (all strategies satisfying $\psi_1$ also satisfy $\psi_2$)
    • (e.g. for assume–guarantee reasoning)
  – time–bounded (finite–horizon) properties
Example – Multi-objective

- **Achievability query**
  - \( P \geq 0.7 \left[ G \neg \text{hazard} \right] \land P \geq 0.2 \left[ GF \text{ goal}_1 \right] \) ? True (achievable)

- **Numerical query**
  - \( P_{\text{max}=?} \left[ GF \text{ goal}_1 \right] \) such that \( P \geq 0.7 \left[ G \neg \text{hazard} \right] \) ? \( \sim 0.2278 \)

- **Pareto query**
  - for \( P_{\text{max}=?} \left[ G \neg \text{hazard} \right] \land P_{\text{max}=?} \left[ GF \text{ goal}_1 \right] \) ?
Example – Multi-objective

\[
\psi_1 = G \neg \text{hazard}
\]

\[
\psi_2 = GF \text{goal}_1
\]

Strategy 1
(deterministic)

\[
\begin{align*}
\text{s}_0 & : \text{east} \\
\text{s}_1 & : \text{south} \\
\text{s}_2 & : - \\
\text{s}_3 & : - \\
\text{s}_4 & : \text{east} \\
\text{s}_5 & : \text{west}
\end{align*}
\]
Example – Multi-objective

Strategy 2
(deterministic)

$s_0 : \text{south}$
$s_1 : \text{south}$
$s_2 : -$  
$s_3 : -$  
$s_4 : \text{east}$  
$s_5 : \text{west}$

$\psi_1 = G \neg \text{hazard}$
$\psi_2 = GF \text{ goal}_1$
Example – Multi-objective

Optimal strategy:
(randomised)

\[ s_0 : 0.3226 : \text{east} \]
\[ 0.6774 : \text{south} \]

\[ s_1 : 1.0 : \text{south} \]
\[ s_2 : - \]
\[ s_3 : - \]
\[ s_4 : 1.0 : \text{east} \]
\[ s_5 : 1.0 : \text{west} \]
Application: Partially satisfiable tasks

- Partially satisfiable task specifications
  - via multi-objective probabilistic model checking [IJCAI'15]
  - e.g. $P_{\text{max}=?} [ \neg \text{zone}_3 \ U (\text{room}_1 \land (F \text{room}_4 \land F \text{room}_5)) ] < 1$

- Synthesise strategies that, in decreasing order of priority:
  - maximise the probability of finishing the task;
  - maximise progress towards completion, if this is not possible;
  - minimise the expected time (or cost) required

- Progress metric constructed from DFA
  - (distance to accepting states, reward for decreasing distance)

- Encode prioritisation using multi-objective queries:
  - $p = P_{\text{max}=?} [ \text{task} ]$
  - $r = \text{multi}(R_{\text{max}=?}^{\text{prog}} [ C ], P_{\geq p} [ \text{task} ])$
  - $\text{multi}(R_{\text{min}=?}^{\text{time}} [ \text{task} ], P_{\geq p} [ \text{task} ] \land R_{\geq r}^{\text{prog}} [ C ])}$
Conclusion

• **Rigorous probabilistic guarantees for robot navigation**
  – formal verification + probabilistic model checking
  – Markov decision processes (MDPs)
  – linear temporal logic (LTL)
  – multi-objective model checking

• **More details**
  – Lacerda/Parker/Hawes. Optimal & Dynamic Planning for Markov Decision Processes with Co-Safe LTL Specifications, IROS'14
  – Lacerda/Parker/Hawes. Optimal Policy Generation for Partially Satisfiable Co-Safe LTL Specifications, IJCAI'15
  – PRISM: [www.prismmodelchecker.org](http://www.prismmodelchecker.org)