Probabilistic verification and synthesis

Marta Kwiatkowska

Department of Computer Science, University of Oxford

KTH, Stockholm, August 2015
Lecture plan

• Course slides and lab session
  – http://www.prismmodelchecker.org/courses/kth15/

• 5 sessions: lectures 9–12noon, labs 2.30–5pm
  – 1 – Introduction
  – 2 – Discrete time Markov chains (DTMCs)
  – 3 – Markov decision processes (MDPs)
  – 4 – LTL model checking for DTMCs/MDPs
  – 5 – Probabilistic timed automata (PTAs)

• For extended versions of this material
  – and an accompanying list of references
  – see: http://www.prismmodelchecker.org/lectures/
# Probabilistic models

<table>
<thead>
<tr>
<th></th>
<th>Fully probabilistic</th>
<th>Nondeterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete time</strong></td>
<td>Discrete-time Markov chains (DTMCs)</td>
<td>Markov decision processes (MDPs)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Simple stochastic games (SMGs)</td>
</tr>
<tr>
<td><strong>Continuous time</strong></td>
<td>Continuous-time Markov chains (CTMCs)</td>
<td>Probabilistic timed automata (PTAs)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Interactive Markov chains (IMCs)</td>
</tr>
</tbody>
</table>
Part 5

Probabilistic Timed Automata
Recall – MDPs

• **Markov decision processes (MDPs)**
  – mix probability and nondeterminism
  – in a state, there is a nondeterministic choice between multiple probability distributions over successor states

- ![MDP Diagram](image)

• **Adversaries**
  – resolve nondeterministic choices based on history so far
  – properties quantify over all possible adversaries
  – e.g. \( P_{<0.1}[\diamond \text{err}] \) – maximum probability of an error is \(< 0.1\)
Real-world protocol examples

• Systems with probability, nondeterminism and real-time
  – e.g. communication protocols, randomised security protocols

• Randomised back-off schemes
  – Ethernet, WiFi (802.11), Zigbee (802.15.4)

• Random choice of waiting time
  – Bluetooth device discovery phase
  – Root contention in IEEE 1394 FireWire

• Random choice over a set of possible addresses
  – IPv4 dynamic configuration (link-local addressing)

• Random choice of a destination
  – Crowds anonymity, gossip-based routing
Overview (Part 5)

- **Time, clocks and zones**
- **Probabilistic timed automata (PTAs)**
  - definition, examples, semantics, time divergence
- **PTCTL: A temporal logic for PTAs**
  - syntax, examples, semantics
- **Model checking for PTAs**
  - the region graph
  - digital clocks
  - zone-based approaches:
    - (i) forwards reachability
    - (ii) backwards reachability
    - (iii) game-based abstraction refinement
- **Costs and rewards**
- **Parameter synthesis**
Real-time systems verification

- Classical model checking
  - labelled transition systems as models
  - CTL as specification notation

- Many systems feature real-time aspects
  - embedded systems
  - in-car and in-flight systems
  - communication protocols
  - controllers
  - etc

- Real-time model checking (e.g. UPPAAL)
  - timed automata as models
  - TCTL as specification notation
Light control example

Modelling...

Spec:
If light is off, press switch once for dimmed light, press switch twice quickly for bright light. Otherwise the light is turned off.
Light control example

Modelling…

Spec:
If light is off, press switch once for dimmed light, press switch twice quickly for bright light. Otherwise the light is turned off.
Light control example

Modelling...

Spec:
If light is off, press switch once for dimmed light, press switch twice quickly for bright light. Otherwise the light is turned off.
Light control example

Modelling...

Spec:
If light is off, press switch once for dimmed light, press switch twice quickly for bright light. Otherwise the light is turned off.
Light control example

Modelling...

Spec:
If light is off, press switch once for dimmed light, press switch twice quickly for bright light. Otherwise the light is turned off.
Light control example

Modelling...

Spec: If light is off, press switch once for dimmed light, press switch twice quickly for bright light. Otherwise the light is turned off.
Light control example

Modelling with time...

Spec:
If light is off, press switch once for dimmed light, press switch twice quickly for bright light. Otherwise the light is turned off.
A timed automaton is a finite graph:

- Finite set of locations
- Finitely many labelled transitions between locations
- Transitions take no time (are instantaneous)
- Automaton can remain in a location for a period of time

Time passage
Continuous, rather than discrete steps
Time elapse
Choice between remaining in location or taking transition

Clocks, here $t$
- real-valued variables
- increase at the same rate as time
- initially $t=0$
- after a period in $\text{Loc}_1$, it is reset to zero
Guards

Guards enable progress

Transitions in timed automata
- can be guarded
- a guard, e.g. $t < 3$, is a constraint on the value of clock $t$
- specifies when the transition is enabled
- i.e. $t = 4$ means precisely

Time elapse
Automaton can remain here forever
Invariants enforce progress

Locations in timed automata
- can have invariants
- i.e. a constraint for remaining in the location
Time, clocks and clock valuations

• **Dense time domain:** non-negative reals $\mathbb{R}_{\geq 0}$
  – from this point on, we will abbreviate $\mathbb{R}_{\geq 0}$ to $\mathbb{R}$

• **Finite set of clocks** $x \in X$
  – variables taking values from time domain $\mathbb{R}$
  – increase at the same rate as real time

• **A clock valuation** is a tuple $v \in \mathbb{R}^X$. Some notation:
  – $v(x)$: value of clock $x$ in $v$
  – $v+t$: time increment of $t$ for $v$
    · $(v+t)(x) = v(x)+t$ $\forall x \in X$
  – $v[Y:=0]$: clock reset of clocks $Y \subseteq X$ in $v$
    · $v[Y:=0](x) = 0$ if $x \in Y$ and $v(x)$ otherwise
Zones (clock constraints)

- Zones (clock constraints) over clocks $X$, denoted $\text{Zones}(X)$:

\[
\zeta ::= x \leq d \mid c \leq x \mid x+c \leq y+d \mid \neg \zeta \mid \zeta \lor \zeta
\]

- where $x, y \in X$ and $c, d \in \mathbb{N}$
- used for both syntax and algorithms

- Some useful classes of zones:
  - **closed**: no strict inequalities (e.g. $x>5$)
  - **convex**: define a convex set, efficient to manipulate

- Can derive:
  - logical connectives, e.g. $\zeta_1 \land \zeta_2 \equiv \neg(\neg \zeta_1 \lor \neg \zeta_2)$
  - strict inequalities, through negation, e.g. $x>5 \equiv \neg(x\leq5)\ldots$
Zones and clock valuations

• A clock valuation $v$ satisfies a zone $\zeta$, written $v \triangleright \zeta$ if
  – $\zeta$ resolves to true after substituting each clock $x$ with $v(x)$

• The semantics of a zone $\zeta \in \text{Zones}(X)$ is the set of clock valuations which satisfy it (i.e. a subset of $\mathbb{R}^X$)
  – NB: multiple zones may have the same semantics
  – e.g. $(x \leq 2) \land (y \leq 1) \land (x \leq y + 2)$ and $(x \leq 2) \land (y \leq 1) \land (x \leq y + 3)$

• We consider only canonical zones
  – i.e. zones for which the constraints are as ‘tight’ as possible
  – $O(|X|^3)$ algorithm to compute (unique) canonical zone \cite{Dil89}
  – allows us to use syntax for zones interchangeably with semantic, set-theoretic operations
  – c-closure, $\text{close}(\zeta,c)$, ignores all constraints which are greater than $c$
Operations on zones – Set theoretic

- Intersection of two zones: $\zeta_1 \cap \zeta_2$
Operations on zones – Set theoretic

- Union of two zones: \( \zeta_1 \cup \zeta_2 \)
Operations on zones – Set theoretic

• Difference of two zones: $\zeta_1 \setminus \zeta_2$
Operations on zones – Clock resets

\[ \zeta[Y:=0] = \{ v[Y:=0] \mid v \triangleright \zeta \} \]

– clock valuations obtained from \( \zeta \) by resetting the clocks in \( Y \)
Operations on zones: Projections

- Forwards diagonal projection
- \( \{ v | \exists t \geq 0 . (v-t) \triangleright \zeta \} \)
  - contains the clock valuations that can be reached from \( \zeta \) by letting time pass
Operations on zones: c–closure

- c–closure: close(ζ, c)
  - ignores all constraints which are greater than c
Overview (Part 5)

• Time, clocks and zones
• **Probabilistic timed automata (PTAs)**
  – definition, examples, semantics, time divergence
• **PTCTL: A temporal logic for PTAs**
  – syntax, examples, semantics
• **Model checking for PTAs**
  – the region graph
  – digital clocks
  – zone-based approaches:
    – (i) forwards reachability
    – (ii) backwards reachability
    – (iii) game-based abstraction refinement
• **Costs and rewards**
• **Parameter synthesis**
• Models a probabilistic real-time communication protocol
  – starts in location \( di \); after between 1 and 2 time units, the protocol attempts to send the data:
    • with probability 0.9 data is sent correctly, move to location \( sr \)
    • with probability 0.1 data is lost, move to location \( si \)
  – in location \( si \), after 2 to 3 time units, attempts to resend
    • correctly sent with probability 0.95 and lost with probability 0.05
Probabilistic timed automata (PTAs)

- Probabilistic timed automata (PTAs)
  - Markov decision processes (MDPs) + real-valued clocks
  - or: timed automata + discrete probabilistic choice
  - model probabilistic, nondeterministic and timed behaviour

- Syntax: A PTA is a tuple \((\text{Loc}, l_{\text{init}}, \text{Act}, X, \text{inv}, \text{prob}, L)\)
  - \text{Loc} is a finite set of locations
  - \(l_{\text{init}} \in \text{Loc}\) is the initial location
  - \text{Act} is a finite set of actions
  - \(X\) is a finite set of clocks
  - \text{inv} \colon \text{Loc} \to \text{Zones}(X)
    is the invariant condition
  - \text{prob} \subseteq \text{Loc} \times \text{Zones}(X) \times \text{Dist(}\text{Loc} \times 2^X\) is the probabilistic transition relation
  - \(L : \text{Loc} \to \text{AP}\) is a labelling function
### Probabilistic transition relation

**Probabilistic edge relation**
- \( \text{prob} \subseteq \text{Loc} \times \text{Zones}(X) \times \text{Act} \times \text{Dist}(\text{Loc} \times 2^X) \)

**Probabilistic transition** \( (l, g, a, p) \in \text{prob} \)
- \( l \) is the source location
- \( g \) is the guard
- \( a \) is the action
- \( p \) target distribution

**Edge** \( (l, g, a, p, l', Y) \)
- from probabilistic edge \( (l, g, a, p) \) where \( p(l', Y) > 0 \)
- \( l' \) is the target location
- \( Y \) is the set of clocks to be reset
PTAs – Behaviour

• A state of a PTA is a pair \((l,v) \in \text{Loc} \times \mathbb{R}^x\) such that \(v \triangleright \text{inv}(l)\)

• A PTAs start in the initial location with all clocks set to zero
  – let \(\_0\) denote the clock valuation where all clocks have value 0

• For any state \((l,v)\), there is nondeterministic choice between making a discrete transition and letting time pass
  – discrete transition \((l,g,a,p)\) enabled if \(v \triangleright g\) and probability of moving to location \(l'\) and resetting the clocks \(Y\) equals \(p(l',Y)\)
  – time transition available only if invariant \(\text{inv}(l)\) is continuously satisfied while time elapses
PTA – Example

PTA:

Example execution:

\[(di, x=0)\]

1.1

\[(di, x=0.1)\]

\[(di, x=1.1)\]

\[(di, x=0)\]

\[(di, x=0.9)\]

\[retry\]

\[x = 0.95\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]

\[retry\]

\[x = 0.05\]

\[send\]

\[x = 0.05\]
PTAs – Formal semantics

• Formally, the semantics of a PTA $P$ is an infinite-state MDP $M_P = (S_P, s_{\text{init}}, \text{Steps}, L_P)$ with:

• States: $S_P = \{ (l,v) \in \text{Loc} \times \mathbb{R}^X \text{ such that } v \triangleright \text{inv}(l) \}$

• Initial state: $s_{\text{init}} = (l_{\text{init}}, 0)$

• Steps: $S_P \rightarrow 2^{(\text{Act} \cup \mathbb{R}) \times \text{Dist}(S)}$ such that $(\alpha, \mu) \in \text{Steps}(l,v)$ iff:
  – (time transition) $\alpha = t \in \mathbb{R}$, $\mu(l,v+t) = 1$ and $v+t' \triangleright \text{inv}(l)$ for all $t' \leq t$
  – (discrete transition) $\alpha = a \in \text{Act}$ and there exists $(l,g,a,p) \in \text{prob}$ such that $v \triangleright g$ and, for any $(l',v') \in S_P$: $\mu(l',v') = \sum_{Y \subseteq X \wedge v[Y:=0]=v'} p(l', Y)$

• Labelling: $L_P(l,v) = L(l)$

actions of MDP $M_p$ are the actions of PTA $P$ or real time delays

multiple resets may give same clock valuation
We restrict our attention to time divergent behaviour
- a common restriction imposed in real-time systems
- unrealisable behaviour (i.e. corresponding to time not advancing beyond a time bound) is disregarded
- also called non-zeno behaviour

For a path \( \omega = s_0(\alpha_0, \mu_0)s_1(\alpha_1, \mu_1)s_2(\alpha_2, \mu_2)\ldots \) in the MDP \( M_P \)
- \( D_\omega(n) \) denotes the duration up to state \( s_n \)
- i.e. \( D_\omega(n) = \sum \{ | \alpha_i | \mid 0 \leq i < n \land \alpha_i \in \mathbb{R} \} \)

A path \( \omega \) is time divergent if, for any \( t \in \mathbb{R}_{\geq 0} \):
- there exists \( j \in \mathbb{N} \) such that \( D_\omega(j) > t \)

Example of non-divergent path:
- \( s_0(1, \mu_0)s_0(0.5, \mu_0)s_0(0.25, \mu_0)s_0(0.125, \mu_0)s_0\ldots \)
PTCTL – Syntax

- **PTCTL**: Probabilistic timed computation tree logic
  - derived from PCTL [BdA95] and TCTL [AD94]

- **Syntax**:
  \[ \phi \ ::= \text{true} \mid a \mid \zeta \mid z. \phi \mid \phi \land \phi \mid \neg \phi \mid \text{P}_{\sim p}[\phi U \phi] \]

- **where**:
  - where \( Z \) is a set of formula clocks, \( \zeta \in \text{Zones}(X \cup Z), \ z \in Z \),
  - \( a \) is an atomic proposition, \( p \in [0,1] \) and \( \sim \in \{<,>,\leq,\geq\} \)

\[ \phi U \phi \text{ is true with probability } \sim p \]

“zone over \( X \cup Z \)”

“freeze quantifier”
PTCTL – Examples

• \( z \cdot P_{>0.99}[\text{packet2unsent} U \text{packet1delivered} \land (z<5)] \)
  – “with probability greater than 0.99, the system delivers packet 1 within 5 time units and does not try to send packet 2 in the meantime”

• \( z \cdot P_{>0.95}[(x\leq3) U (z=8)] \)
  – “with probability at least 0.95, the system clock \( x \) does not exceed 3 before 8 time units elapse”

• \( z \cdot P_{\leq0.1}[\text{failure} \lor (z\leq60)] \)
  – “the system fails after the first 60 time units have elapsed with probability at most 0.01”
• Let \((l,v) \in S_p\) and \(\varepsilon \in \mathbb{R}^Z\) be a formula clock valuation.

Combined clock valuation of \(v\) and \(\varepsilon\) satisfies \(\zeta\):

- \((l,v),\varepsilon \models a\) \iff \(a \in L(l,v)\)
- \((l,v),\varepsilon \models \zeta\) \iff \(v,\varepsilon \triangleright \zeta\)
- \((l,v),\varepsilon \models z.\phi\) \iff \((l,v),\varepsilon[z:=0] \models \phi\)
- \((l,v),\varepsilon \models \phi_1 \land \phi_2\) \iff \((l,v),\varepsilon \models \phi_1\) and \((l,v),\varepsilon \models \phi_2\)
- \((l,v),\varepsilon \models \neg \phi\) \iff \((l,v),\varepsilon \models \phi\) is false
- \((l,v),\varepsilon \models P_{\neg p}[\psi]\) \iff \(\Pr^{A,p}_{(l,v)}\{ \omega \in \text{Path}^A(l,v) \mid \omega,\varepsilon \models \psi \} \sim p\)

For all adversaries \(A \in \text{Adv}_{M_p}\)

The probability of a path satisfying \(\psi\) meets \(\sim p\) for all divergent adversaries.
PTCTL – Semantics of until

- Let $\omega$ be a path in $M_p$ and $\mathcal{E}$ be a formula clock valuation
  - $\omega, \mathcal{E} \models \psi$ satisfaction of $\psi$ by $\omega$, assuming $\mathcal{E}$ initially

- $\omega, \mathcal{E} \models \phi_1 \mathbf{U} \phi_2$ if and only if
  - there exists $i \in \mathbb{N}$ and $t \in D_\omega(i+1)-D_\omega(i)$ such that
    - $\omega(i)+t, \mathcal{E}+(D_\omega(i)+t) \models \phi_2$
    - $\forall t' \leq t . \omega(i)+t', \mathcal{E}+(D_\omega(i)+t') \models \phi_1 \lor \phi_2$
    - $\forall j<i . \forall t' \leq D_\omega(j+1)-D_\omega(j) . \omega(j)+t', \mathcal{E}+(D_\omega(j)+t') \models \phi_1 \lor \phi_2$

- Condition “$\phi_1 \lor \phi_2$” different from PCTL and CSL
  - usually $\phi_2$ becomes true and $\phi_1$ is true until this point
  - difference due to the density of the time domain
  - to allow for open intervals use disjunction $\phi_1 \lor \phi_2$
  - for example consider $x \leq 5 \mathbf{U} x > 5$ and $x < 5 \mathbf{U} x \geq 5$
For simplicity, in some cases, we just consider probabilistic reachability, rather than full PTCTL model checking:
- i.e. min/max probability of reaching a set of target locations
- can also encode time-bound reachability (with extra clock)

Still captures a wide range of properties:
- probabilistic reachability: “with probability at least 0.999, a data packet is correctly delivered”
- probabilistic invariance: “with probability 0.875 or greater, the system never aborts”
- probabilistic time-bound reachability: “with probability 0.01 or less, a data packet is lost within 5 time units”
- bounded response: “with probability 0.99 or greater, a data packet will always be delivered within 5 time units”
Overview (Part 5)

• Time, clocks and zones
• Probabilistic timed automata (PTAs)
  – definition, examples, semantics, time divergence
• PTCTL: A temporal logic for PTAs
  – syntax, examples, semantics
• Model checking for PTAs
  – the region graph
  – digital clocks
  – zone-based approaches:
    – (i) forwards reachability
    – (ii) backwards reachability
    – (iii) game-based abstraction refinement
• Costs and rewards
• Parameter synthesis
PTA model checking – Summary

• Several different approaches developed
  – basic idea: reduce to the analysis of a finite-state model
  – in most cases, this is a Markov decision process (MDP)

• Region graph construction [KNSS02]
  – shows decidability, but gives exponential complexity

• Digital clocks approach [KNPS06]
  – (slightly) restricted classes of PTAs
  – works well in practice, still some scalability limitations

• Zone-based approaches:
  – (preferred approach for non-probabilistic timed automata)
  – forwards reachability [KNSS02]
  – backwards reachability [KNSW07]
  – game-based abstraction refinement [KNP09c]
The region graph

- **Region graph construction for PTAs** [KNSS02]
  - adapts region graph construction for timed automata [ACD93]
  - partitions PTA states into a **finite** set of **regions**
  - based on notion of clock equivalence
  - construction is also dependent on PTCTL formula

- **For a PTA $P$ and PTCTL formula $\phi$**
  - construct a **time-abstract, finite-state MDP** $R(\phi)$
  - translate PTCTL formula $\phi$ to PCTL formula $\phi'$
  - $\phi$ is preserved by region equivalence
  - i.e. $\phi$ holds in a state of $M_P$ if and only if $\phi'$ holds in the corresponding state of $R(\phi)$
  - model check $R(\phi)$ using standard methods for MDPs
The region graph – Clock equivalence

- **Regions** are sets of clock equivalent clock valuations

- **Some notation:**
  - let $c$ be largest constant appearing in PTA or PTCTL formula
  - let $\lfloor t \rfloor$ denotes the integral part of $t$
  - $t$ and $t'$ agree on their integral parts if and only if
    1. $\lfloor t \rfloor = \lfloor t' \rfloor$
    2. $t$ and $t'$ are both integers or neither is an integer

- **The clock valuations $v$ and $v'$ are clock equivalent ($v \cong v'$) if:**
  - for all clocks $x \in X$, either:
    - $v(x)$ and $v'(x)$ agree on their integral parts
    - $v(x) > c$ and $v'(x) > c$
  - for all clock pairs $x, x' \in X$, either:
    - $v(x) - v(x')$ and $v'(x) - v'(x')$ agree on their integral parts
    - $v(x) - v(x') > c$ and $v'(x) - v'(x') > c$
• Example regions (for 2 clocks $x$ and $y$)

$x=1 \land y=2$

$x<y \land 1<x<2 \land 1<y<2$

$x=y \land 0<x<1$

$y=1 \land 2<x<3$
Region graph – Clock equivalence

• Fundamental result: if \( v \equiv v' \), then \( v \triangleright \zeta \iff v' \triangleright \zeta \)
  – it follows that \( r \triangleright \zeta \) is well defined for a region \( r \)

• \( r' \) is the successor region of \( r \), written \( \text{succ}(r) = r' \), if
  – for each \( v \in r \), there exists \( t > 0 \) such that \( v + t \in r' \)
    and \( v + t' \in r \cup r' \) for all \( t' < t \)
The region graph

- The region graph MDP is \((S_R, s_{init}, \text{Steps}_R, L_R)\) where...

  - the set of states \(S_R\) comprises pairs \((l, r)\) such that \(l\) is a location and \(r\) is a region over \(X \cup Z\)
  - the initial state is \((l_{init}, 0)\)
  - the set of actions is \(\{\text{succ}\} \cup \text{Act}\)
    - \(\text{succ}\) is a unique action denoting passage of time
  - the probabilistic transition function \(\text{Steps}_R\) is defined as:
    - \(S_R \times 2^{\{\text{succ}\} \cup \text{Act}} \times \text{Dist}(S_R)\)
    - \((\text{succ}, \mu) \in \text{Steps}_R(l, r)\) iff \(\mu(l, \text{succ}(r)) = 1\)
    - \((a, \mu) \in \text{Steps}_R(l, r)\) if and only if \(\exists (l, g, a, p) \in \text{prob}\) such that
      \[
      r \triangleright g \quad \text{and, for any} \quad (l', r') \in S_R: \quad \mu(l', r') = \sum_{Y \subseteq X \land r[Y := 0] = r'} p(l', Y)
      \]
  - the labelling is given by: \(L_R(l, r) = L(l)\)
Region graph – Example

- PTCTL formula: $z.P_{\neg p} [ \text{true} \ U (sr<4) ]$

\[
\begin{align*}
\text{Node} & \quad \text{Label} & \quad \text{Prob.} \\
\text{di, } x=z=0 & \quad \text{send} & \quad 0.9 \\
\text{di, } x=z=0 & \quad \text{retry} & \quad 0.1 \\
\text{di, } 0<x=z<1 & \quad \text{success} & \quad 0.9 \\
\text{di, } 0<x=z<1 & \quad \text{retry} & \quad 0.1 \\
\text{di, } x=z=1 & \quad \text{success} & \quad 0.95 \\
\text{di, } 1<x=z<2 & \quad \text{success} & \quad 0.05 \\
\end{align*}
\]
Region graph construction

- **Region graph**
  - useful for establishing *decidability* of model checking
  - or proving *complexity* results for model checking algorithms

- **But…**
  - the number of regions is *exponential* in the number of clocks and the size of largest constant
  - so model checking based on this is extremely expensive
  - and so not implemented (even for timed automata)

- **Improved approaches based on:**
  - digital clocks
  - zones (unions of regions)
Digital clocks

- **Simple idea:** Clocks can only take integer (digital) values
  - i.e. time domain is $\mathbb{N}$ as opposed to $\mathbb{R}$
  - based on notion of $\varepsilon$-digitisation [HMP92]

- **Only applies to a restricted class of PTAs; zones must be:**
  - **closed** – no strict inequalities (e.g. $x > 5$)

- **Digital clocks semantics yields a finite-state MDP**
  - state space is a subset of $\text{Local} \times \mathbb{N}^X$, rather than $\text{Local} \times \mathbb{R}^X$
  - clocks bounded by $c_{\text{max}}$ (max constant in PTA and formula)
  - then use standard techniques for finite-state MDPs
Example – Digital clocks

**MDP:** (digital clocks)

- \((\text{di}, x = z = 0)\) with probability 0.9
- \((\text{di}, x = z = 1)\) with probability 0.1
- \((\text{di}, x = z = 2)\) with probability 0.9

- \((\text{sr}, x = 0 \land z = 1)\) with probability 0.1
- \((\text{si}, x = 0 \land z = 1)\)
- \((\text{sr}, x = 0 \land z = 2)\)

- \((\text{si}, x = 1 \land z = 2)\)
- \((\text{si}, x = 2 \land z = 3)\)

**PTA:**

- \(x \leq 2\)
- \(x \geq 1\)
- \(x = 0\)
- \(x = 3\)

- **di**
  - send
  - \(x := 0\) with probability 0.9
  - \(x := 0\) with probability 0.05

- **sr**
  - true
  - \(x := 0\) with probability 0.95
  - \(x := 0\) with probability 0.05

- **si**
  - \(x \leq 3\)
  - \(x \geq 2\)
  - retry
  - \(x := 0\) with probability 0.1
  - \(x := 0\) with probability 0.05

- **PTA:**
  - : :
Digital clocks

• Digital clocks approach preserves:
  – minimum/maximum reachability probabilities
  – a subset of PTCTL properties
  – (no nesting, only closed zones in formulae)
  – only works for the initial state of the PTA
  – (but can be extended to any state with integer clock values)

• In practice:
  – translation from PTA to MDP can often be done manually
  – (by encoding the PTA directly into the PRISM language)
  – automated translations exist
  – many case studies, despite “closed” restriction

• Problem: can lead to very large MDPs
  – alleviated partially by efficient symbolic model checking
Zone-based approaches

- An alternative is to use zones to construct an MDP

- Conventional symbolic model checking relies on computing
  - $\text{post}(S')$ the states that can be reached from a state in $S'$ in a single step
  - $\text{pre}(S')$ the states that can reach $S'$ in a single step

- Extend these operators to include time passage
  - $d\text{post}[e](S')$ the states that can be reached from a state in $S'$ by traversing the edge $e$
  - $t\text{post}(S')$ the states that can be reached from a state in $S'$ by letting time elapse
  - $d\text{pre}[e](S')$ the states that can reach $S'$ by traversing the edge $e$
  - $t\text{pre}(S')$ the states that can reach $S'$ by letting time elapse
Zone-based approaches

• **Symbolic states** \((l, \zeta)\) where
  - \(l \in \text{Loc} \) (location)
  - \(\zeta\) is a zone over PTA clocks and formula clocks
  - generally fewer zones than regions

\[
\text{tpost}(l, \zeta) = (l, \neg \zeta \land \text{inv}(l))
\]
  - \(\neg \zeta\) can be reached from \(\zeta\) by letting time pass
  - \(\neg \zeta \land \text{inv}(l)\) must satisfy the **invariant** of the location \(l\)

\[
\text{tpre}(l, \zeta) = (l, \neg \zeta \land \text{inv}(l))
\]
  - \(\neg \zeta\) can reach \(\zeta\) by letting time pass
  - \(\neg \zeta \land \text{inv}(l)\) must satisfy the **invariant** of the location \(l\)
Zone-based approaches

• For an edge $e = (l,g,a,p,l',Y)$ where
  – $l$ is the source
  – $g$ is the guard
  – $a$ is the action
  – $l'$ is the target
  – $Y$ is the clock reset

• $d_{post}[e](l,\zeta) = (l', (\zeta \land g)[Y:=0])$
  – $\zeta \land g$ satisfy the guard of the edge
  – $(\zeta \land g)[Y:=0]$ reset the clocks $Y$

• $d_{pre}[e](l',\zeta') = (l, [Y:=0] \zeta' \land (g \land inv(l)))$
  – $[Y:=0] \zeta'$ the clocks $Y$ were reset
  – $[Y:=0] \zeta' \land (g \land inv(l))$ satisfied guard and invariant of $l$
Forwards reachability

- Based on the operation $\text{post}[e](l,\zeta) = t\text{post}(d\text{post}[e](l,\zeta))$
  - $(l',v') \in \text{post}[e](l,\zeta)$ if there exists $(l,v) \in (l,\zeta)$ such that after traversing edge $e$ and letting time pass one can reach $(l',v')$

- Forwards algorithm (part 1)
  - start with initial state $S_F = \{t\text{post}((l_{\text{init}},0))\}$ then iterate
    for each symbolic state $(l,\zeta) \in S_F$ and edge $e$
    add $\text{post}[e](l,\zeta)$ to $S_F$
  - until set of symbolic states $S_F$ does not change

- To ensure termination need to take $c$–closure of each zone encountered ($c$ is the largest constant in the PTA)
Forwards reachability

- **Forwards algorithm (part 2)**
  - construct finite state MDP \((S_F, (l_{\text{init}}, O), \text{Steps}_F, L_F)\)

- states \(S_F\) (returned from first part of the algorithm)
- \(L_F(l, \zeta) = L(l)\) for all \((l, \zeta) \in S_F\)
- \(\mu \in \text{Steps}_F(l, \zeta)\) if and only if
  - there exists a probabilistic edge \((l, g, a, p)\) of PTA
  - such that for any \((l', \zeta') \in Z:\)

\[
\mu(l', \zeta') = \sum_{\text{edges}(p) \land \text{post}[e](l, \zeta) = (l', \zeta')} \left| p(l', X) \right|
\]

summation over all the edges of \((l, g, a, p)\) such that applying post to \((l, \zeta)\) leads to the symbolic state \((l', \zeta')\)
Forwards reachability – Example

PTA:

\[ l_0 \xrightarrow{0.5} l_1, \text{x:=0} \]
\[ l_1 \xrightarrow{0.5} l_0, \text{y:=0} \]
\[ l_1 \xrightarrow{0.5} l_3, \text{x=0\land y=1} \]
\[ l_3 \xrightarrow{0.5} l_2, \text{x=0\land y=0} \]

MDP:

\[ (l_0, x\leq y) \]
\[ (l_0, x=y) \]
\[ (l_3, x=y) \]
Forwards reachability – Limitations

• Problem reduced to analysis of finite-state MDP, but...

• Only obtain upper bounds on maximum probabilities
  – caused by when edges are combined

• Suppose \( \text{post}[e_1](l, \zeta) = (l_1, \zeta_1) \) and \( \text{post}[e_2](l, \zeta) = (l_2, \zeta_2) \)
  – where \( e_1 \) and \( e_2 \) from the same probabilistic edge

• By definition of post
  – there exists \((l, v_i) \in (l, \zeta)\) such that a state in \((l_i, \zeta_i)\) can be reached by traversing the edge \( e_i \) and letting time pass

• Problem
  – we combine these transitions but are \((l, v_1)\) and \((l, v_2)\) the same?
  – may not exist states in \((l, \zeta)\) for which both edges are enabled
Forwards reachability – Example

• Maximum probability of reaching $l_3$ is 0.5 in the PTA
  – for the left branch need to take the first transition when $x=1$
  – for the right branch need to take the first transition when $x=0$
• However, in the forwards reachability graph probability is 1
  – can reach $l_3$ via either branch from $(l_0, x=y)$

PTA:

```
\begin{center}
\begin{tikzpicture}
  \node (l0) at (0,0) {$l_0$};
  \node (l1) at (1,1) {$l_1$};
  \node (l3) at (2,2) {$l_3$};
  \node (l2) at (3,1) {$l_2$};

  \path[->,thick]
  (l0) edge node[above] {$y:=0$} (l1)
  (l0) edge node[below] {$x:=0$} (l2)
  (l0) edge node {$0.5$} (true)
  (l1) edge node[above] {$x=0 \land y=1$} (l3)
  (l2) edge node[below] {$x=0 \land y=0$} (l3)
  (true) edge node[below] {$0.5$} (l3)
  (l1) edge node[above] {$0.5$} (l3)
  (l2) edge node[below] {$0.5$} (l3)
\end{tikzpicture}
\end{center}
```

MDP:

```
\begin{center}
\begin{tikzpicture}
  \node (l0) at (0,0) {$l_0$};
  \node (l3) at (1,1) {$l_3$};

  \path[->,thick]
  (l0) edge node[above] {$x=0 \land y=1$} (l3)
  (l0) edge node[below] {$x=0 \land y=0$} (l3)
  (l0) edge node[above] {$x=0 \land y=0$} (l3)
  (true) edge node[below] {$0.5$} (l3)
  (l1) edge node[above] {$0.5$} (l3)
  (l2) edge node[below] {$0.5$} (l3)
\end{tikzpicture}
\end{center}
```
An alternative zone-based method: backwards reachability

- state-space exploration in opposite direction, from target to initial states; uses pre rather than post operator

Basic ideas: (see [KNSW07] for details)

- construct a finite-state MDP comprising symbolic states
- need to keep track of branching structure and take conjunctions of symbolic states if necessary
- MDP yields maximum reachability probabilities for PTA
- for min. probs, do graph-based analysis and convert to max.

Advantages:
- gives (exact) minimum/maximum reachability probabilities
- extends to full PTCTL model checking

Disadvantage:
- operations to implement are expensive, limits applicability
- (requires manipulation of non-convex zones)
Overview (Part 5)

- Time, clocks and zones
- Probabilistic timed automata (PTAs)
  - definition, examples, semantics, time divergence
- PTCTL: A temporal logic for PTAs
  - syntax, examples, semantics
- Model checking for PTAs
  - the region graph
  - digital clocks
  - zone-based approaches:
    - (i) forwards reachability
    - (ii) backwards reachability
    - (iii) game-based abstraction refinement
- Costs and rewards
- Parameter synthesis
Abstraction

- Very successful in (non-probabilistic) formal methods
  - essential for verification of large/infinite-state systems
  - hide details irrelevant to the property of interest
  - yields smaller/finite model which is easier/feasible to verify
  - loss of precision: verification can return “don’t know”
- Construct abstract model of a concrete system
  - e.g. based on a partition of the concrete state space
  - an abstract state represents a set of concrete states

- Automatic generation of abstractions using refinement
  - start with a simple coarse abstraction; iteratively refine
Abstraction of MDPs

- Abstraction increases degree of nondeterminism
  - i.e. minimum probabilities are lower and maximums higher

- We construct abstractions of MDPs using stochastic games
  - yields lower/upper bounds for min/max probabilities
Abstraction refinement

• Consider (max) difference between lower/upper bounds
  – gives a quantitative measure of the abstraction’s precision

• If the difference (“error”) is too great, refine the abstraction
  – a finer partition yields a more precise abstraction
  – lower/upper bounds can tell us where to refine (which states)
  – (memoryless) strategies can tell us how to refine
Abstraction–refinement loop

- Quantitative abstraction–refinement loop for MDPs

- Refinements yield strictly finer partition

- Guaranteed to converge for finite models

- Guaranteed to converge for infinite models with finite bisimulation
Abstraction refinement for PTAs

• Model checking for PTAs using abstraction refinement

Initial abstraction from forwards reachability

Splitting of zones (DBMs)

Guaranteed convergence for any $\epsilon \geq 0$

Abstraction computed and stored using zones (DBMs)
Abstraction refinement for PTAs

• Computes reachability probabilities in PTAs
  – minimum or maximum, exact values ("error" \( \epsilon = 0 \))
  – also time-bounded reachability, with extra clock

• Integrated in PRISM (development release)
  – PRISM modelling language extended with clocks
  – implemented using DBMs

• In practice, performs very well
  – faster than digital clocks or backwards on large example set
  – (sometimes by several orders of magnitude)
  – handles larger PTAs than the digital clocks approach
Costs and rewards

• Like other models, we can define a reward structure \((\rho, \iota)\) for a probabilistic timed automaton

• \(\rho : \text{Loc} \rightarrow \mathbb{R}_{\geq 0}\) location reward function
  – \(\rho(l)\) is the rate at which the reward is accumulated in location \(l\)

• \(\iota : \text{Act} \rightarrow \mathbb{R}_{\geq 0}\) action reward function
  – \(\iota(a)\) is the reward associated with performing the action \(a\)

• Generalises notion for uniformly priced timed automata

• A useful special case is the elapsed time
  – \(\rho(l) = 1\) for all locations \(l \in \text{Loc}\)
  – \(\iota(a) = 0\) for all actions \(a \in \text{Act}\)
Expected reachability

• **Expected reachability:**
  – min./max. expected cumulated reward until some set of states (locations) is reached

• **Example properties**
  – “the maximum expected time until a data packet is delivered”
  – “the minimum expected number of retransmissions before the message is correctly delivered”
  – “the maximum expected number of lost messages within the first 200 seconds”

• **Model checking**
  – digital clocks semantics preserves expected reachability
  – so can use existing MDP reward model checking techniques
  – zone-based approaches solved recently [FORMATS 2015]
Summary

• **Probabilistic timed automata (PTAs)**
  - combine probability, nondeterminism, real-time
  - well suited for e.g. for randomised communication protocols
  - MDPs + clocks (or timed automata + discrete probability)
  - extension with continuous distributions exists, but model checking only approximate

• **PTCTL: Temporal logic for properties of PTAs**
  - but many useful properties expressible with just reachability

• **PTA model checking**
  - region graph: decidability results, exponential complexity
  - digital clocks: simple and effective, some scalability issues
  - forwards reachability: only upper bounds on max. prob.s
  - backwards reachability: exact results but often expensive
  - abstraction refinement using stochastic games: performs well
• **New features:**
  1. parametric model checking
  2. parameter synthesis
  3. strategy synthesis
  4. stochastic multi-player games
  5. *real-time: probabilistic timed automata (PTAs) [CAV 2015]*

• **Further new additions:**
  – enhanced statistical model checking
    (approximations + confidence intervals, acceptance sampling)
  – efficient CTMC model checking (fast adaptive uniformisation)
  – benchmark suite & testing functionality
  – [www.prismmodelchecker.org](http://www.prismmodelchecker.org)

  – **Beyond PRISM…**
Modelling PTAs in PRISM

- **Probability + nondeterminism + real-time**
  - timed automata + discrete probabilistic choice, or...
  - probabilistic automata + real-valued clocks

- **PTA example**: message transmission over faulty channel

![Diagram of a PTA example]

**States**
- locations + data variables

**Transitions**
- guards and action labels

**Real-valued clocks**
- state invariants, guards, resets

**Probability**
- discrete probabilistic choice
Modelling PTAs in PRISM

- **PRISM modelling language**
  - textual language, based on guarded commands

```plaintext
pta
const int N;
module transmitter
  s : [0..3] init 0;
  tries : [0..N+1] init 0;
  x : clock;
  invariant (s=0 ⇒ x≤2) & (s=1 ⇒ x≤5) endinvariant
  [send] s=0 & tries≤N & x≥1
    → 0.9 : (s'=3)
    + 0.1 : (s'=1) & (tries'=tries+1) & (x'=0);
  [retry] s=1 & x≥3 → (s' =0) & (x' =0);
  [quit] s=0 & tries>N → (s’ =2);
endmodule
rewards “energy” (s=0) : 2.5; endrewards
```
Modelling PTAs in PRISM

- PRISM modelling language
  - textual language, based on guarded commands

```plaintext
pta
const int N;
module transmitter
  s : [0..3] init 0;
  tries : [0..N+1] init 0;
  x : clock;
  invariant (s=0 ⇒ x≤2) & (s=1 ⇒ x≤5) endinvariant

  [send] s=0 & tries≤N & x≥1
    → 0.9 : (s’=3)
    + 0.1 : (s’=1) & (tries’=tries+1) & (x’=0);
  [retry] s=1 & x≥3 → (s’ =0) & (x’ =0);
  [quit] s=0 & tries>N → (s’ =2);
endmodule
rewards “energy” (s=0) : 2.5; endrewards
```

Basic ingredients:
- modules
- variables
- commands
Modelling PTAs in PRISM

- **PRISM modelling language**
  - textual language, based on guarded commands

```plaintext
pta
const int N;
module transmitter
  s : [0..3] init 0;
  tries : [0..N+1] init 0;
  x : clock;
  invariant (s=0 ⇒ x≤2) & (s=1 ⇒ x≤5) endinvariant
  [send] s=0 & tries≤N & x≥1 → 0.9 : (s’=3)
  + 0.1 : (s’=1) & (tries’=tries+1) & (x’=0);
  [retry] s=1 & x≥3 → (s’ =0) & (x’ =0);
  [quit] s=0 & tries>N → (s’ =2);
endmodule
rewards “energy” (s=0) : 2.5; endrewards
```

**Basic ingredients:**
- modules
- variables
- commands

**New for PTAs:**
- clocks
- invariants
- guards/resets
Modelling PTAs in PRISM

- **PRISM modelling language**
  - textual language, based on guarded commands

```plaintext
pta
const int N;
module transmitter
  s : [0..3] init 0;
  tries : [0..N+1] init 0;
  x : clock;
  invariant (s=0 ⇒ x≤2) & (s=1 ⇒ x≤5) endinvariant
[send] s=0 & tries≤N & x≥1
  → 0.9 : (s'=3)
  + 0.1 : (s'=1) & (tries'=tries+1) & (x'=0);
[retry] s=1 & x≥3 → (s' =0) & (x' =0);
[quit] s=0 & tries>N → (s' =2);
endmodule
rewards “energy” (s=0) : 2.5; endrewards
```

**Basic ingredients:**
- modules
- variables
- commands

**New for PTAs:**
- clocks
- invariants
- guards/resets

**Also:**
- rewards (i.e. costs, prices)
- parallel composition
Model checking PTAs in PRISM

- **Properties for PTAs:**
  - min/max probability of reaching X (within time T)
  - min/max expected cost/reward to reach X
    (for “linearly-priced” PTAs, i.e. reward gain linear with time)

- **PRISM has two different PTA model checking techniques...**

- **“Digital clocks” – conversion to finite-state MDP**
  - preserves min/max probability + expected cost/reward/price
  - (for PTAs with closed, diagonal-free constraints)
  - efficient, in combination with PRISM’s symbolic engines

- **Quantitative abstraction refinement**
  - zone-based abstractions of PTAs using stochastic games
  - provide lower/upper bounds on quantitative properties
  - automatic iterative abstraction refinement
PRISM: Recent & new developments

- **New features:**
  1. parametric model checking
  2. parameter synthesis
  3. strategy synthesis
  4. stochastic multi-player games
  5. real-time: probabilistic timed automata (PTAs) [CAV 2015]

- **Further new additions:**
  - enhanced statistical model checking (approximations + confidence intervals, acceptance sampling)
  - efficient CTMC model checking (fast adaptive uniformisation)
  - benchmark suite & testing functionality
  - [www.prismmodelchecker.org](http://www.prismmodelchecker.org)

- Beyond PRISM...
Case study: Cardiac pacemaker

- **Develop model-based framework**
  - timed automata model for pacemaker software [Jiang et al]
  - hybrid heart models in Simulink, adopt synthetic ECG model (non-linear ODE) [Clifford et al]

- **Properties**
  - (basic safety) maintain 60–100 beats per minute
  - (advanced) detailed analysis of energy usage, plotted against timing parameters of the pacemaker
  - parameter synthesis: find values for timing delays that optimise energy usage
Optimal timing delays problem

- Optimal timing delay synthesis for timed automata [EMSOFT2014][HSB 2015]
- The parameter synthesis problem solved is
  - given a parametric network of timed I/O automata, set of controllable and uncontrollable parameters, CMTL property \( \phi \) and length of path \( n \)
  - find the optimal controllable parameter values, for any uncontrollable parameter values, with respect to an objective function \( O \), such that the property \( \phi \) is satisfied on paths of length \( n \), if such values exist
- Consider family of objective functions
  - maximise volume, minimise energy
- Discretise parameters, assume bounded integer parameter space and path length
  - decidable but high complexity (high time constants)
Optimal probability of timing delays

- Previously, no nondeterminism and no probability in the model considered
- Consider parametric probabilistic timed automata (PPTA),
  - e.g. guards of the form $x \leq b$,
- Can we synthesise optimal timing parameters to optimise the reachability probability?
- Semi-algorithm [RP 2014]
  - exploration of parametric symbolic states, i.e. location, time zone and parameter valuations
  - forward exploration only gives upper bounds on maximum probability (resp. lower for minimum)
  - but stochastic game abstraction yields the precise solution…
- Implementation in progress
Quantitative verification – Trends

- Being ‘younger’, generally lags behind conventional verification
  - much smaller model capacity
  - compositional reasoning in infancy
  - automation of model extraction/adaptation very limited

- Tool usage on the increase, in academic/industrial contexts
  - real-time verification/synthesis in embedded systems
  - probabilistic verification in security, reliability, performance

- Shift towards greater automation
  - specification mining, model extraction, synthesis, verification, ...

- But many challenges remain!
Acknowledgements

• My group and collaborators in this work
• Project funding
  – ERC, EPSRC, Microsoft Research
  – Oxford Martin School, Institute for the Future of Computing

• See also
  – VERIWARE www.veriware.org
  – PRISM www.prismmodelchecker.org
• You are welcome to visit Oxford!
• PhD scholarships, postdocs in verification and synthesis, and more
Thank you for your attention

More info here:
www.prismmodelchecker.org