Advances in Quantitative Verification for Ubiquitous Computing

Marta Kwiatkowska
University of Oxford

ICTAC 2013, Shanghai

Where are computers?

Once upon a time, back in the 1980s…

Smartphones, tablets…

Access to services:
Email, banking, shopping, directions, …

Personalised monitoring:
GPS/GPRS tracking
Accelerometer, pedometer, …
Air quality

Autonomous systems…

Intelligent transport:

Self-parking cars
Driverless cars
Search and rescue
Unmanned missions
…

House appliances, networked…

Internet of Things

Home network
Internet-enabled
Remote control
Smart energy management

House appliances, networked…

Enabled by service-based systems…

Internet of Things

Home network
Internet-enabled
Remote control
Energy management

Ubiquitous computing

- Computing without computers
- (also known as Pervasive Computing or Internet of Things
  - enabled by wireless technology and cloud computing)
- Populations of sensor-enabled computing devices that are
  - embedded in the environment, or even in our body
  - sensors for interaction and control of the environment
  - software controlled, can communicate
  - operate autonomously, unattended
  - devices are mobile, handheld or wearable
  - miniature size, limited resources, bandwidth and memory
  - organised into communities
- Unstoppable technological progress
  - smaller and smaller devices, more and more complex scenarios…

Perspectives on ubiquitous computing

- Technological: calm technology [Weiser 1993]
  - "The most profound technologies are those that disappear. They weave themselves into everyday life until they are indistinguishable from it."

- Usability: 'everyware' [Greenfield 2008]
  - Hardware/software evolved into 'everyware': household appliances that do computing

- Scientific: "Ubicomp can empower us, if we can understand it" [Milner 2008]
  - "What concepts, theories and tools are needed to specify and describe ubiquitous systems, their subsystems and their interaction?"

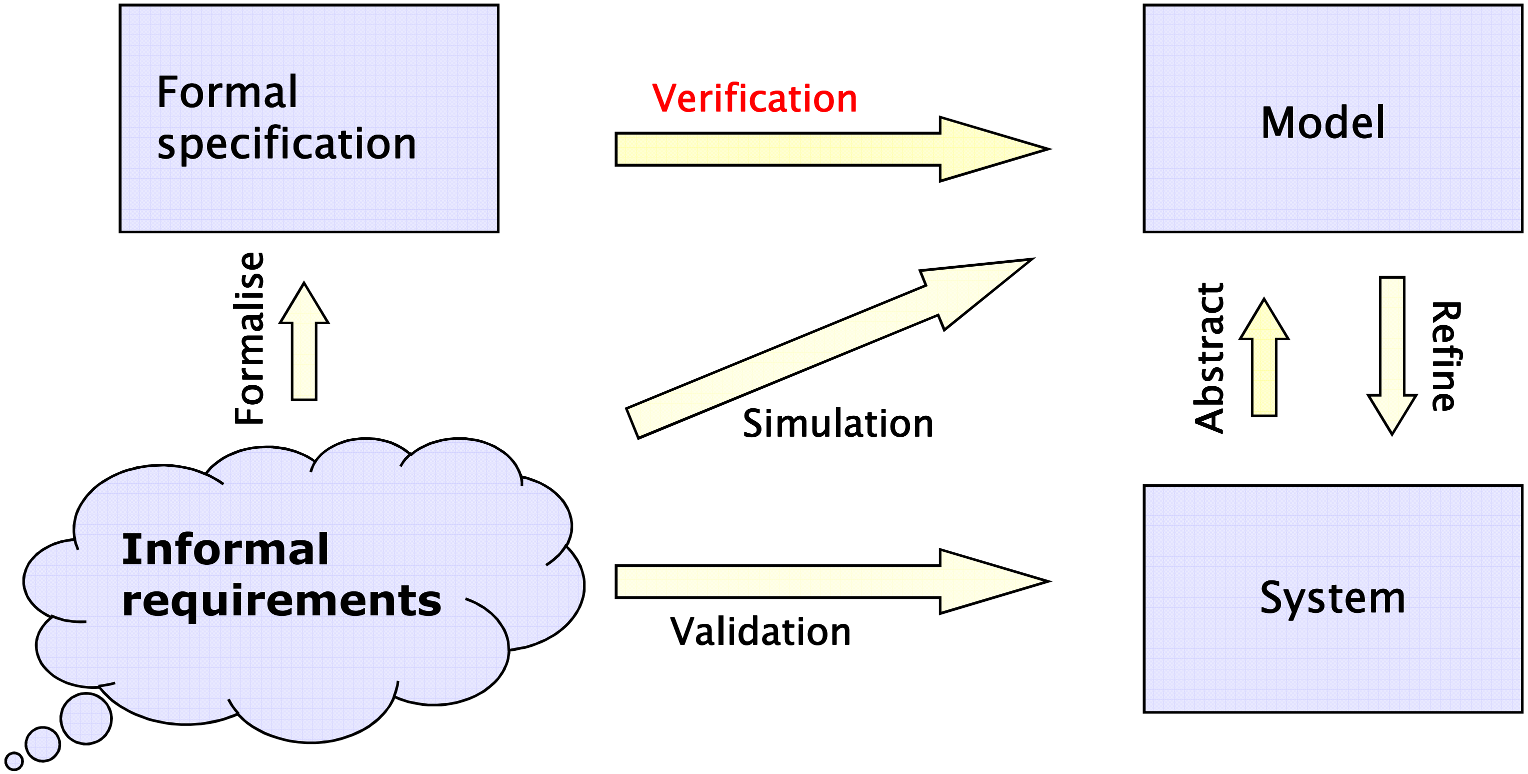
- This lecture concerns verification methodology
  - emphasises practical, algorithmic techniques and industrially-relevant tools

Software quality assurance

- Software is a critical component
  - embedded software failure costly and life endangering
- Need quality assurance methodologies
  - model-based development
  - rigorous software engineering
  - software product lines
- Use formal techniques to produce guarantees for:
  - safety, reliability, performance, resource usage, trust, …
  - (safety) “probability of failure to raise alarm is tolerably low”
  - (reliability) “the smartphone will never execute the financial transaction twice”
- Focus on automated, tool-supported methodologies
  - automated verification via model checking
  - quantitative verification

Rigorous software engineering

- Verification and validation
  - Derive model, or extract from software artefacts
  - Verify correctness, validate if fit for purpose

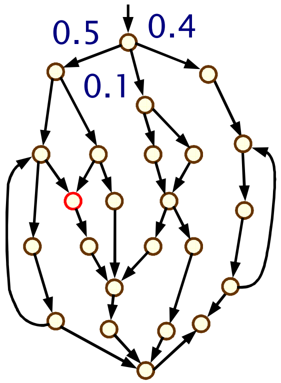


Quantitative verification now

Automatic verification (aka model checking) of quantitative properties of probabilistic system models

System

Probabilistic model
e.g. Markov chain



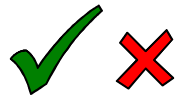
$P_{<0.01}$ [$F^{\leq t}$ fail]

System requirements

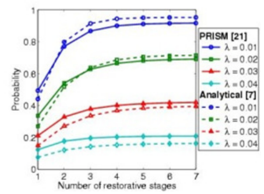
Probabilistic temporal logic specification
e.g. PCTL

Probabilistic model checker
e.g. PRISM

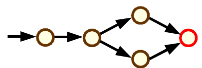
Result



Quantitative results



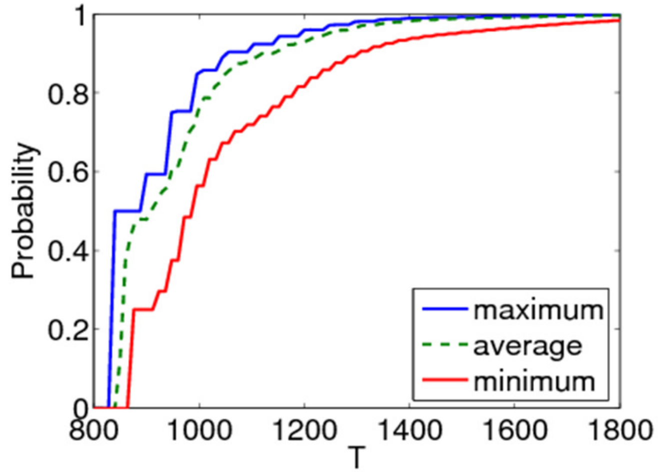
Counter-example



Why quantitative verification?

- Real Ubicomp software/systems are quantitative:
  - Real-time aspects
    - hard/soft time deadlines
  - Resource constraints
    - energy, buffer size, number of unsuccessful transmissions, etc
  - Randomisation, e.g. in distributed coordination algorithms
    - random delays/back-off in Bluetooth, Zigbee
  - Uncertainty, e.g. communication failures/delays
    - prevalence of wireless communication

- Analysis “quantitative” & “exhaustive”
  - strength of mathematical proof
  - best/worst-case scenarios, not possible with simulation
  - identifying trends and anomalies



Historical perspective

- First algorithms proposed in 1980s
  - [Vardi, Courcoubetis, Yannakakis, …]
  - algorithms [Hansson, Jonsson, de Alfaro] & first implementations
- 2000: tools ETMCC (MRMC) & PRISM released
  - PRISM: efficient extensions of symbolic model checking [Kwiatkowska, Norman, Parker, …]
  - ETMCC (now MRMC): model checking for continuous-time Markov chains [Baier, Hermanns, Haverkort, Katoen, …]
- Now mature area, of industrial relevance
  - successfully used by non-experts for many application domains, but full automation and good tool support essential
    - distributed algorithms, communication protocols, security protocols, biological systems, quantum cryptography, planning…
  - genuine flaws found and corrected in real-world systems

Quantitative probabilistic verification

- What's involved
  - specifying, extracting and building of quantitative models
  - graph-based analysis: reachability + qualitative verification
  - numerical solution, e.g. linear equations/linear programming
  - typically computationally more expensive than the non-quantitative case
- The state of the art
  - fast/efficient techniques for a range of probabilistic models
  - feasible for models of up to 10^7 states (10^{10} with symbolic)
  - extension to probabilistic real-time systems
  - abstraction refinement (CEGAR) methods
  - assume-guarantee compositional verification
  - statistical model checking
  - tool support exists and is widely used, e.g. PRISM, MRMC

Quantitative properties

- Probabilistic properties
  - $P_{\leq 0.01}$ [F "fail"] – "the probability of a failure is at most 0.01"
  - $P_{=?}$ [$G^{\leq 0.02}$!"deploy" {"crash"}{max}] – "the maximum probability of an airbag failing to deploy within 0.02s, from any possible crash scenario"
- Reward/cost-based properties
  - $R_{\{\text{"time"}\}=?}$ [F "end"] – "expected algorithm execution time"
  - $R_{\{\text{"energy"}\}\max=?}$ [$C^{\leq 7200}$] – "worst-case expected energy consumption during the first 2 hours"
- Multi-objective properties
  - $P_{\leq 0.01}$ [F "fail"] $\wedge$ $R_{\{\text{"time"}\}\leq 10}$ [F "end"] – "probability of failing is no greater than 0.01 and expected algorithm execution time is less or equal than 10s"

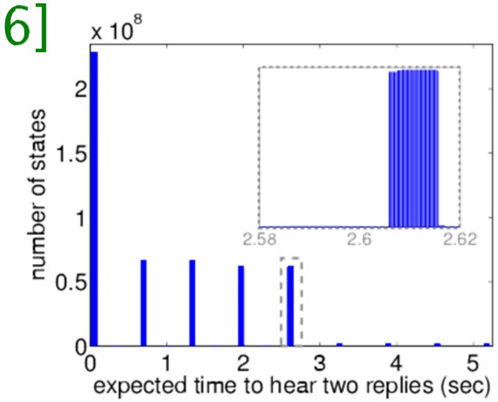
Tool support: PRISM

- PRISM: Probabilistic symbolic model checker [CAV11]
  - developed at Birmingham/Oxford University, since 1999
  - free, open source software (GPL), runs on all major OSs
- Support for:
  - models: DTMCs, CTMCs, MDPs, PTAs, SMGs, …
  - properties: PCTL, CSL, LTL, PCTL*, costs/rewards, rPATL, …
- Features:
  - simple but flexible high-level modelling language
  - user interface: editors, simulator, experiments, graph plotting
  - multiple efficient model checking engines (e.g. symbolic)
  - New! strategy synthesis, stochastic game models (SMGs) for collaborative protocols, parametric models

- See: http://www.prismmodelchecker.org/

Quantitative verification in action

- Bluetooth device discovery protocol [STTT06]
  - frequency hopping, randomised delays
  - low-level model in PRISM, based on detailed Bluetooth reference documentation
  - numerical solution of 32 Markov chains, each approximately 3 billion states
  - identified worst-case time to hear one message



- Microgrid demand management protocol [TACAS12]
  - designed for households to actively manage demand while accessing a variety of energy sources
  - found and fixed a flaw in the protocol, due to lack of punishment for selfish behaviour
  - implemented in PRISM-games

The challenges of ubiquitous computing

- Autonomous behaviour: electronic agents make decisions and act independently of humans, e.g. search and rescue
- Constrained resources: low power, processor speed and memory capacity, intermittent connectivity
- Adaptiveness: systems have to adapt to changing requirements in predictable fashion
- Communities of agents: need to model cooperation, competition and resource sharing, necessitating game-theoretic approaches
- Monitoring and control of physical processes (cyber-physical systems): needed in autonomous transport, robotic planning, implantable medical devices such as pacemakers, etc
- Interfacing with the natural world: biosensing and DNA/molecular computation have important applications in disease detection and drug delivery

This lecture…

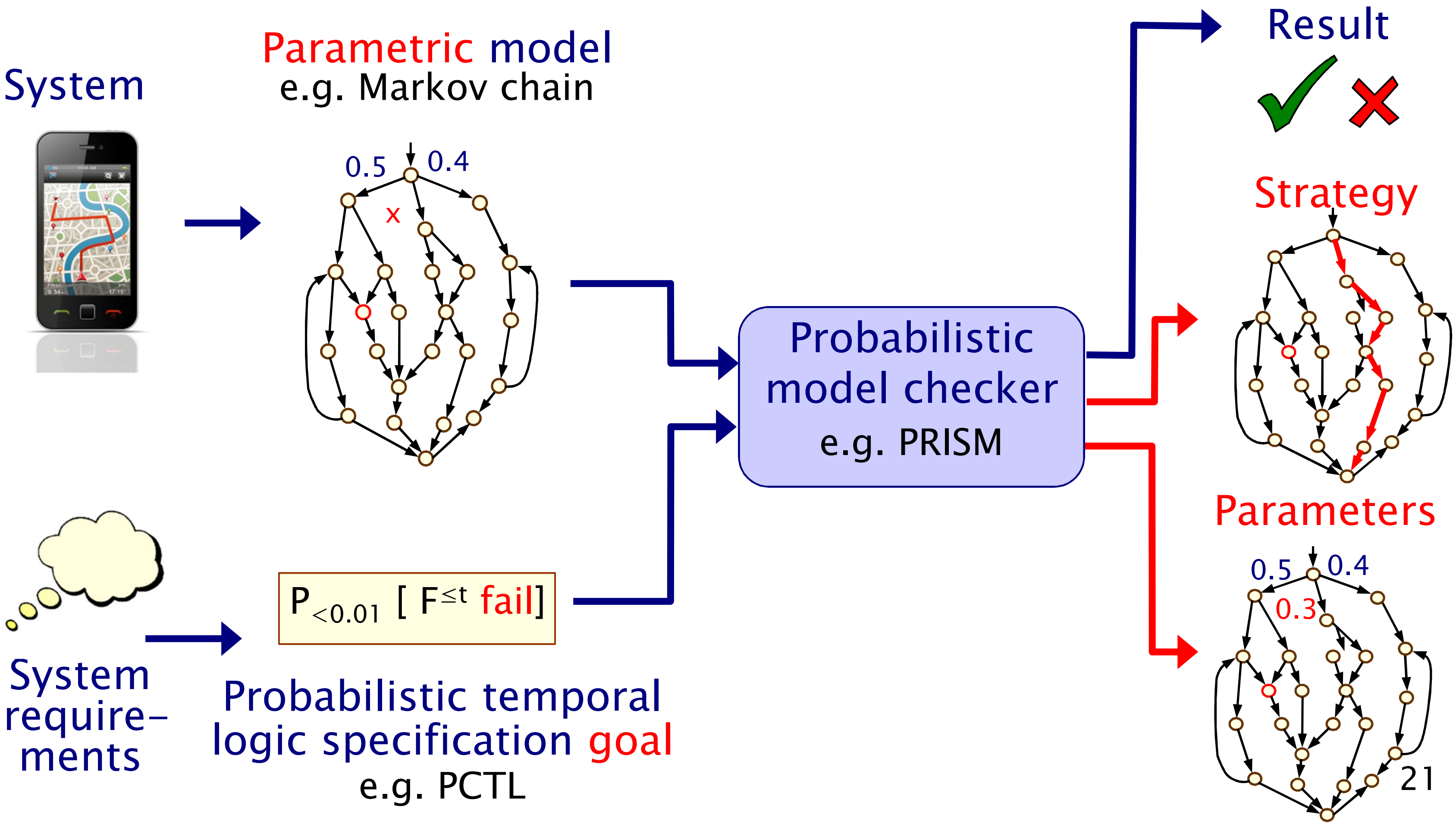
- Show applications of quantitative verification in ubiquitous computing, highlighting new directions under development

- Majority of research to date has focused on
  - scalability and performance of verification algorithms
  - extending expressiveness of models and logics
  - industrially-relevant case studies

- In this lecture, we focus on three research questions:
  1. How to guarantee correct autonomous behaviour?
  2. How to handle constrained resources?
  3. How to ensure predictable adaptation?

From quantitative verification to synthesis

Automatic synthesis of correct-by-construction strategies and models from quantitative properties/goals



The focus of this lecture

Quantitative verification is not powerful enough!

1. How to guarantee correct autonomous behaviour?
   - shift from verification to controller synthesis from quantitative temporal specifications
2. How to handle constrained resources?
   - enable parametric models and determine values that ensure the satisfaction of a given property
3. How to ensure predictable adaptation?
   - shift from offline to quantitative runtime verification

Aim to describe the above directions

- each employing PRISM, at Oxford or elsewhere
- formulating novel frameworks or algorithms

1. Autonomous behaviour

- Research question:
  - How to guarantee correct autonomous behaviour?

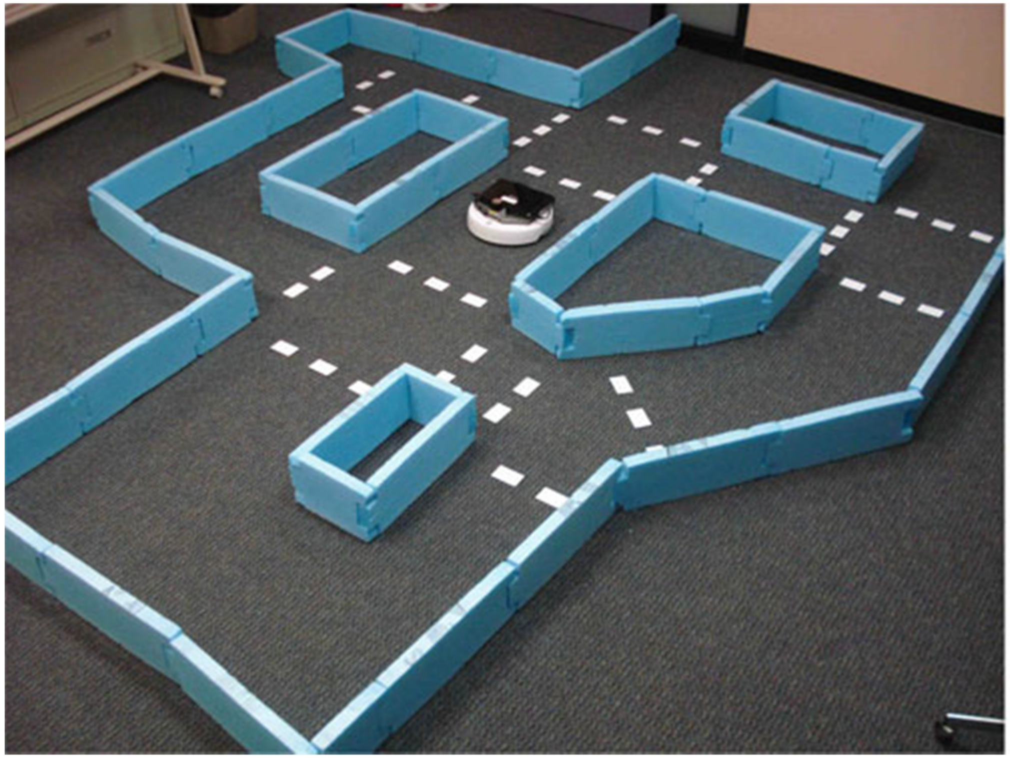
- Many Ubicomp scenarios!
  - robotics
  - search and rescue
  - autonomous vehicles
  - unmanned missions

- Approach
  - Specify (quantitative) goals in temporal logic
  - Derive controllers that guarantee the satisfaction of the goals

- In other words, verification meets robotics and control…

The (simplified) setting

- Robotic motion planning
  [Belta et al]



- Partitioned (indoor) environment, hence a transition system
- Motion primitives: Go Left, Go Straight, …
- Specify goals in temporal logic: “Reach A while avoiding B”

Quantitative goals

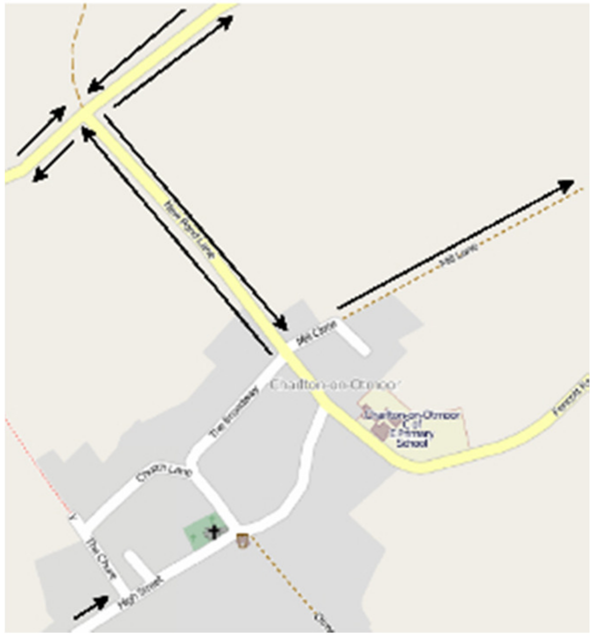
- Why probability?
  - need to consider sensor noise, hence motion primitives may have probabilistic outcomes
  - assume simplified setting of perfect information (achievable with RFID tags)
  - obtain a discrete Markov decision process
  - decorate with rewards, to also consider e.g. expected energy or time
- Specify goals in temporal logic PCTL with rewards
  - $P_{=max?}$ [(“S” $\vee$ (“R” $\wedge$ “M”)) U “D”] – reach “Destination” by driving through either “Safe” regions or through “Relatively safe” regions only if “Medical supply” is available there
  - $R_{\{\text{“time”}\}=min?}$ [F “D”] – eventually reach “Destination” while minimising time to get there
  - also their combinations

Quantitative controller synthesis

- The problem statement is as follows
  - Given a Markov decision process and a PCTL formula ϕ, find the controller strategy that maximises the probability of satisfying ϕ
  - (similar for minimisation and expected rewards)
- Solution
  - compute the maximum probability by e.g. linear programming or value iteration
  - 'read off' the optimal strategy
  - more complicated when strategies are history dependent…
- In [Belta et al], applied to compute controllers for iRobot and safe vehicle control
  - PRISM used for simple (single formula) goals, and otherwise algorithms extended
  - guarantees validated experimentally

Autonomous urban driving

- Inspired by DARPA challenge, in [QEST13][MFCS13] we consider a more general problem
  - represent map data as a stochastic game, with environment able to select hazards
  - express goals as conjunctions of probabilistic and reward properties
  - e.g. "maximise probability of avoiding hazards and minimise time to reach destination"



- Solution (PRISM-games)
  - synthesise a probabilistic strategy to achieve the multi-objective goal
  - enable the exploration of trade-offs between subgoals
- Applied to synthesise driving strategies for English villages
  - being developed in PRISM-games

2. Constrained resources

- Research question
  - How to handle constrained resources, such as low power, memory and processor capacity of processors, RFIDs, etc?
- Aims
  - develop a sound understanding of the impact of constrained resources on performance of critical functions of mobile devices
  - find optimal parameter values
- Approach
  - Devise generic/parametric models and analyse their performance wrt realistic parameter values
  - Synthesise (optimal) parameter values to guarantee that the property is satisfied

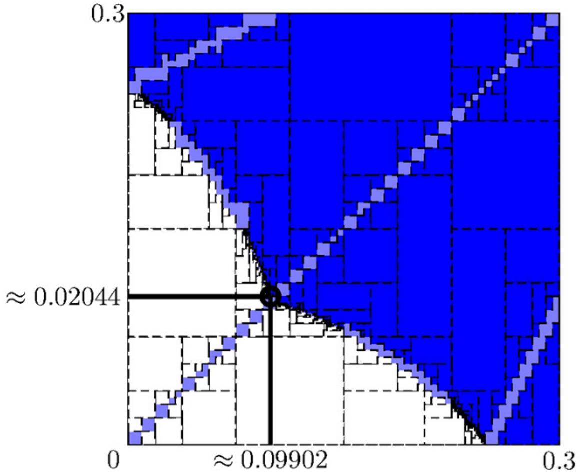
Motivating example

- Quantitative analysis of a Certified Email Delivery (CEMD) protocol of [Basagiannis et al]
- Smartphones increasingly often used to access sensitive services
  - cryptographic protocols necessary
  - yet low power/capacity processors used in HSDPA (High Speed Downlink Packet Access) mobile environments
  - variable Bit Error Rate, hence noise affects transmissions
- Need to consider impact of the setting on critical functions
  - duthors derive generic continuous-time Markov chain model
  - analyse performance wrt realistic parameters, showing considerable impact on reliability
- Proposed methodological enhancement (new in PRISM):
  - devise a parametric model, then obtain optimal parameter values given reliability goals

Parametric models in PRISM

- Can specify models in parametric form [TASE13]
  - parameters expressed as unevaluated constants
  - e.g. const double x;
  - transition probabilities specified as expressions over parameters, e.g. 0.5 + x
- Properties are given in PCTL, with parameter constants
  - new construct constfilter (min, x1*x2, prop)
  - filters over parameter values, rather than states
- Determine parameter valuations to guarantee satisfaction of given properties
- Two methods implemented in PRISM ('explicit' engine)
  - constraints-based approach is a reimplementation of PARAM 2.0 [Hahn et al]
  - sampling-based approaches are new implementation

Case study: parametric models



Checking if minimal exp. number of attacks >= 20

Property constfilter(min,…,$R_{\{\text{“attacks”}\}>=20}$ [F “end”])

Model (network virus) has 809 states, $\epsilon = 0.05$

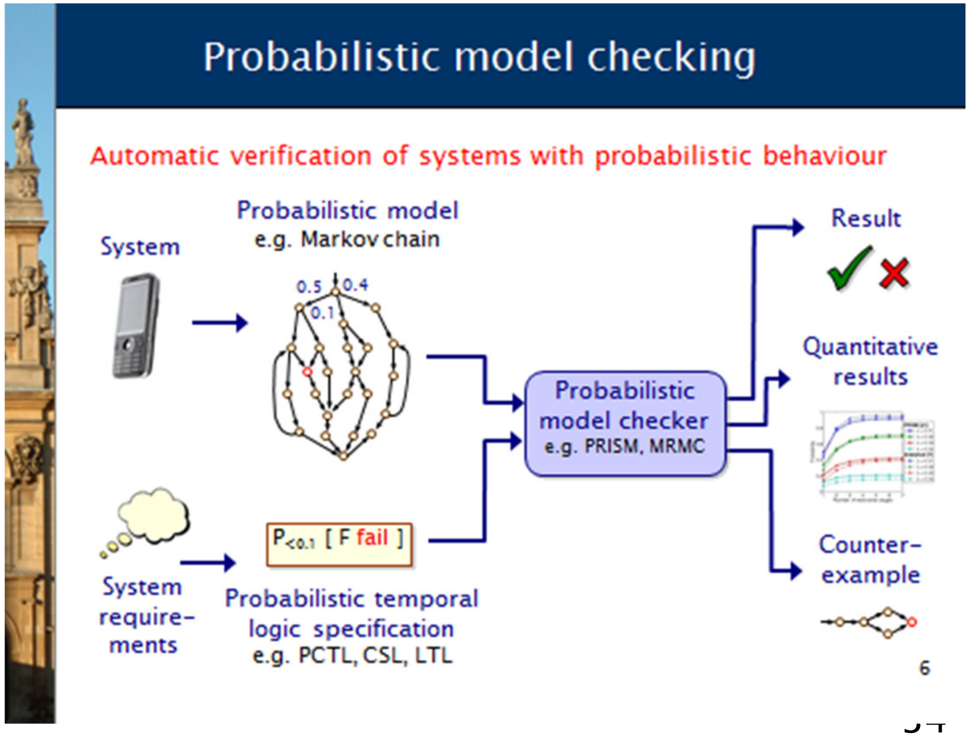
Optimal value found in 2mins, showing optimal parameter values

3. Adaptiveness

- Research question
  - How to ensure predictable adaptation?
- Service-based systems, e.g. cloud computing, are essential for Ubicomp
  - online commerce, healthcare, banking, …
- Predictability needed in presence of
  - component failure
  - environmental uncertainty
  - changing requirements
- Approach
  - Monitor , verify and enforce at runtime
  - Steer computation away from danger states

Offline quantitative verification

- Derive quantitative formal models:
  - probabilistic, annotated with time, energy cost, …
- Use temporal logics to specify:
  - reliability, performance, resource usage, …
- Apply quantitative/ probabilistic verification tools at design time
  - offline, to ensure correctness before deployment
- As good as ability to anticipate problems…
- What if requirements change at runtime?



DOI:10.1145/2330667.2330686

Continually verify self-adaptation decisions taken by critical software in response to changes in the operating environment.

BY RADU CALINESCU, CARLO GHEZZI, MARTA KWIATKOWSKA, AND RAFFAELA MIRANDOLA

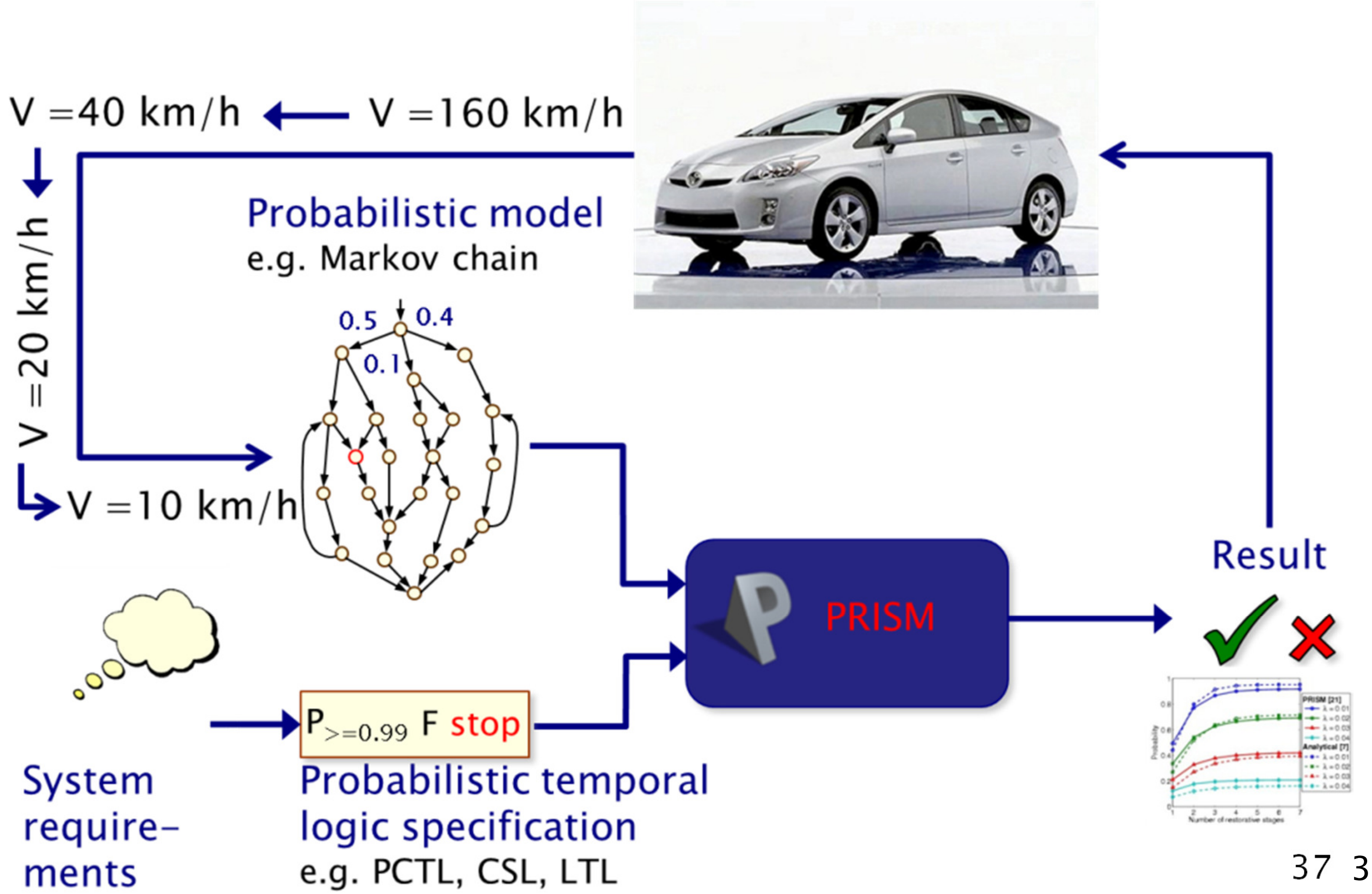
Self-Adaptive Software Needs Quantitative Verification at Runtime

» key insights

- **Human activity increasingly relies on software being able to make self-adaptation decisions on the fly.**
- **Offline approaches to verifying correctness before software deployment must be accompanied by continual online verification of the software's self-adaptation decisions.**
- **Quantitative verification at runtime supports continual re-verification of key requirements of self-adaptive software.**

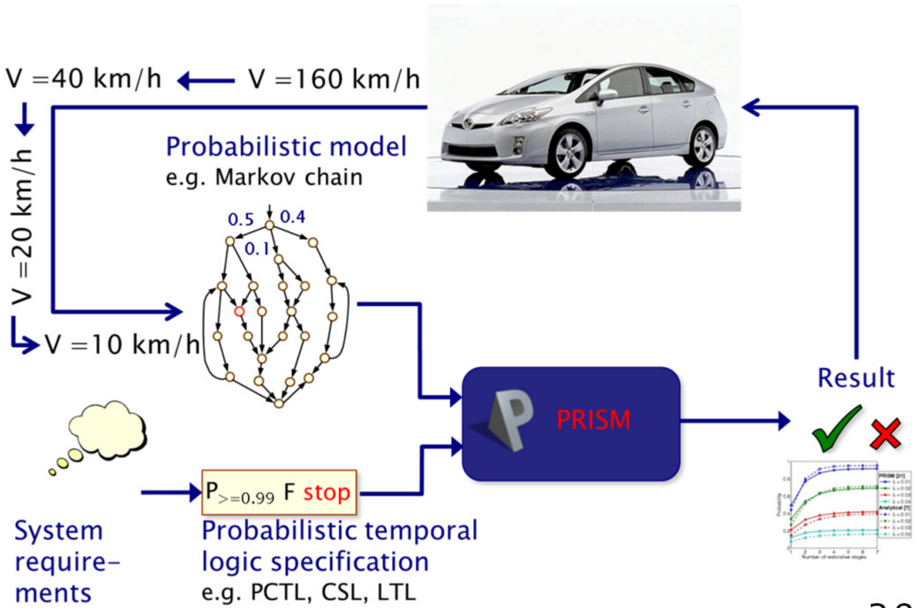
Online quantitative verification…

… for adaptive systems, to capture failure and uncertainty



Online quantitative verification

- Implemented in QoSMOS framework [TSE2011]
  - enables natural language specification, runtime monitoring and enforcement, and learning from history
  - uses PRISM to select optimal services
  - incremental model construction and verification techniques, i.e. re-using previous results [DSN11][RV12]



Summing up…

- Brief overview of three new directions in quantitative verification
  - highlighted a growing shift from automated verification to correct-by-construction synthesis
  - demonstrated usefulness of quantitative verification methodology, particularly as implemented in PRISM
  - new techniques and frameworks
- Many challenges remain
  - scalability of the techniques
  - synthesis of models
  - more expressive models, e.g. cyber-physical systems
- More challenges not covered in this lecture
  - implantable medical devices, collaboration and competition, biosensing, …

References

- Autonomous behaviour
  - T. Chen, M. Kwiatkowska, A. Simaitis and C. Wiltsche. Synthesis for Multi-Objective Stochastic Games: An Application to Autonomous Urban Driving. In Proc. QEST'13.
  - T. Chen, V. Forejt, M. Kwiatkowska, A. Simaitis and C. Wiltsche. On Stochastic Games with Multiple Objectives. In Proc. MFCS'13.
- Constrained resources
  - T. Chen, E. Hahn, T. Han, M. Kwiatkowska, H. Qu and L. Zhang. Model Repair for Markov Decision Processes. In Proc. TASE'13.
- Adaptiveness
  - R. Calinescu, C. Ghezzi, M. Kwiatkowska and R. Mirandola.S elf-adaptive Software Needs Quantitative Verification at Runtime. Communications of the ACM, 55(9), pages 69-77, ACM, 2012.
- See also
  - M. Kwiatkowska, G. Norman and D. Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. CAV 2011: 585-591.

Acknowledgements

- My group and collaborators in this work
- Collaborators who contributed to theoretical and practical PRISM development
- External users of, and contributors to, PRISM
- Project funding
  - ERC, EPSRC
  - Oxford Martin School, Institute for the Future of Computing
- See also
  - VERIWARE www.veriware.org
  - PRISM www.prismmodelchecker.org