Probabilistic model checking in practice: Case studies with PRISM

Marta Kwiatkowska

School of Computer Science

THE UNIVERSITY OF BIRMINGHAM

www.cs.bham.ac.uk/~mzk

www.cs.bham.ac.uk/~dxp/prism

QAPL, 3rd April 2005

Overview

- Probabilistic model checking
  - Why needed?
  - What does it involve?
- The PRISM model checker
  - About the tool
  - Main functionality
- Case studies
  - Self-stabilisation algorithms
  - Molecular reactions
  - Contract signing protocols
  - Bluetooth device discovery
- Challenges for future

With thanks to…

- Main collaborators on probabilistic model checking
  - Gethin Norman, Dave Parker, Jeremy Sproston, Christel Baier, Roberto Segala, Michael Huth, Luca de Alfaro, Joost-Pieter Katoen, Antonio Pacheco
- PRISM model checker implementation
  - Dave Parker, Andrew Hinton, Rashid Mehmood, Yi Zhang, Hakan Younes, Stephen Gilmore, Michael Goldsmith, Conrado Daws, Fuzhi Wang
- Case studies
  - Vitaly Shmatikov, Gethin Norman, Marie Duflot, Jeremy Sproston, Sandeep Shukla, Rajesh Gupta, Carroll Morgan, Annabelle McIver
- And many more…

Ubiquitous computing: the trends…

- Devices, ever smaller
  - Laptops, phones, PDAs, …
  - Sensors, motes, …
- Networking, wireless, wired & global
  - Mobile ad hoc
  - Wireless everywhere
  - Internet everywhere
  - Global connectivity
- Systems/software
  - Self-configuring
  - Self-organising
  - Bio-inspired
  - Autonomous
  - Adaptive
  - Context-aware




Ubiquitous computing: users expect…

- …assurance of
  - safety
  - correctness
  - performance
  - reliability

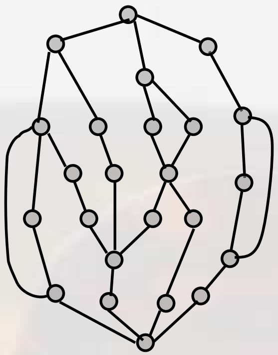
- For example:
  - Is my e-savings account secure?
  - Can someone bluesnarf from my phone?
  - How fast is the communication from my PDA to printer?
  - Is my mobile phone energy efficient?
  - Is the operating system reliable?
  - Is the protocol fault tolerant?

Probability helps

- In distributed co-ordination algorithms
  - As a symmetry breaker
    - "leader election is eventually resolved with probability 1"
  - In fault-tolerant schemes
    - "the message will be delivered to all nodes with high probability"
- When modelling uncertainty in the environment
  - To quantify failures, express soft deadlines, QoS
    - "the chance of shutdown is at most 0.1%"
    - "the probability of a frame delivered within 5ms is at least 0.91"
  - To quantify environmental factors in decision support
    - "the expected cost of reaching the goal is 100"
- When analysing system performance
  - To quantify arrivals, service, etc, characteristics
    - "in the long run, mean waiting time in a lift queue is 30 sec"

Verification via model checking…

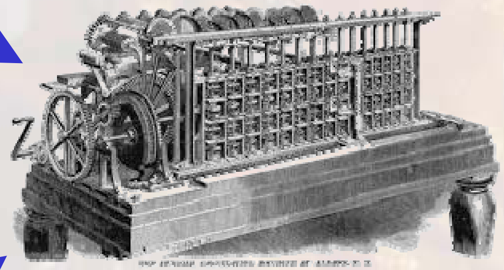
or falsification?



The model

send $\rightarrow \Diamond$deliver

Temporal logic specification



Model Checker

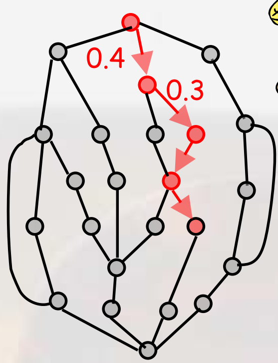
or

Error trace

Line 5: …
Line 21: …
Line 15: …
…
Line 27: …
Line 45: …

Probabilistic model checking…

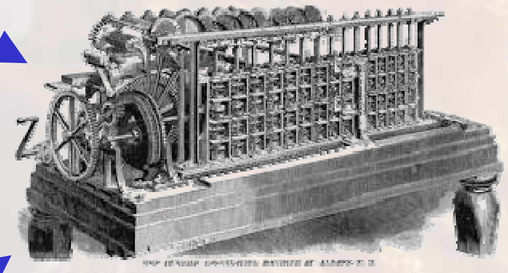
in a nutshell



Probabilistic model

$\text{send} \rightarrow \mathbf{P}_{\geq 0.9}(\Diamond \text{deliver})$

Probabilistic temporal logic specification



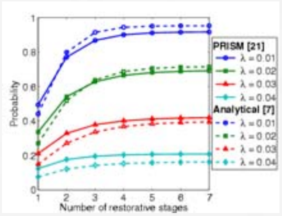
Probabilistic Model Checker

or

or

The probability

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
…
State 12: 0
State 13: 0.1245



Probabilistic model checking inputs…

- Models
  - discrete time Markov chains (DTMCs)
  - continuous time Markov chains (CTMCs)
  - Markov decision processes (MDPs)
  - (currently indirectly) probabilistic timed automata (PTAs)

- (Yes/No) temporal logic specification languages
  - Probabilistic temporal logic PCTL (for DTMCs/MDPs)
  - Continuous Stochastic Logic CSL (for CTMCs)
  - Probabilistic timed computation tree logic PTCTL (for PTAs)

- Quantitative specification language variants
  - Probability values for logics PCTL/CSL/PTCTL (for all models)
  - Extension with expectation operator (for all)
  - Extension with costs/rewards (for all)

Probabilistic model checking involves…

- Construction of models:
  - discrete and continuous Markov chains (DTMCs/CTMCs)
  - Markov decision processes (MDPs), and
  - probabilistic timed automata (PTAs)

- Implementation of probabilistic model checking algorithms
  - graph-theoretical algorithms, combined with
    - (probabilistic) reachability
    - qualitative model checking (for 0/1 probability)
  - numerical computation – iterative methods
    - quantitative model checking (plot probability values, expectations, rewards, steady-state, etc, for a range of parameters)
    - exhaustive, unlike simulation

The PRISM probabilistic model checker

- Approach
  - Based on symbolic, BDD-based techniques
  - Multi-Terminal BDDs, first algorithm [ICALP'97]
  - Hybrid combination of symbolic and explicit vector representation, efficient for CTMCs
- History
  - First public release September 2001, ~7 years development
  - Substantial improvements to functionality, efficiency and model size capability (> 10^{10} for CTMCs, higher for other models)
- Funding
  - EPSRC, several projects including ongoing projects on compositionality, mobility extension and parallelisation
  - DTI/QinetiQ, project FORWARD
  - British Council, collaboration with Germany, France and Portugal

The PRISM tool: overview

- Functionality
  - Implements temporal logic probabilistic model checking
  - Construction of models: discrete and continuous Markov chains (DTMCs/CTMCs), and Markov decision processes (MDPs)
  - Modelling language: probabilistic guarded commands
  - Probabilistic temporal logics: PCTL and CSL
  - Extension with costs/rewards, expectation operator

- Underlying computation combines graph-theoretical algorithms
  - Reachability, qualitative model checking, BDD-based

  with numerical computation – iterative methods
  - Linear equation system solution - Jacobi, Gauss-Seidel, ...
  - Uniformisation (CTMCs)
  - Dynamic programming (MDPs)
  - Explicit and symbolic (MTBDDs, etc.)

PRISM modelling language

- Simple, state-based language for DTMCs/CTMCs/MDPs
  - based on Reactive Modules [Alur/Henzinger]
- Basic components:
  - modules (system components, parallel composition)
  - variables (finite-state, typed)
  - guarded commands (probabilistic, action-labelled)

[send] (s=2) -> p_{loss} : (s'=3)&(lost'=lost+1) + (1-p_{loss}) : (s'=4);

action	guard	probability	update	probability	update
[send]	(s=2)	p_{loss}	(s'=3)&(lost'=lost+1)	(1-p_{loss})	(s'=4)

More on PRISM modelling language...

- Other features:
  - synchronisation on action labellings
  - process algebra style specifications
    - parallel composition: P1 ||| P2, P1 |[a,b]| P2, P1 || P2
    - action hiding/renaming: P/{a}, P{a<-b}

  - import of PEPA models
  - state-dependent probabilities/rates
  - global variables
  - macros

  - import of CSP+probability models

PRISM property specifications

- PCTL/CSL (true/false) formula examples:
  - P≥1 [true U terminate]
    "the algorithm eventually terminates successfully with probability 1"
  - P<0.001 [true U≤100 error]
    "the probability of the system reaching an error state within 100 time units is less than 0.001"
  - down => P>0.75 [!fail U[1,2.5] up]
    "when shutdown occurs, the probability of system recovery between 1 and 2.5 hours, without further failures occurring, is greater than 0.75"

- Can also write query formulae:
  - P=? [true U≤10 terminate]
    "what is the probability that the algorithm terminates successfully within 10 time units?"

PRISM technicalities

- Augment states and transitions with real-valued rewards
  - Instantaneous rewards, e.g. “concentration of reactant”
  - Cumulative rewards, state- and transition-based, e.g. “power consumed”, “messages lost”
- Support for “experiments”
  - e.g. P=? [true U<=T error] for N=1..5,T=1..100
- GUI implementation
  - integrated editor for PRISM language
  - automatic graph plotting
- (Ongoing) Simulator and sampling-based model checking
  - allows to “excute” the model step-by-step or randomly
  - avoids state-space explosion, trading off accuracy

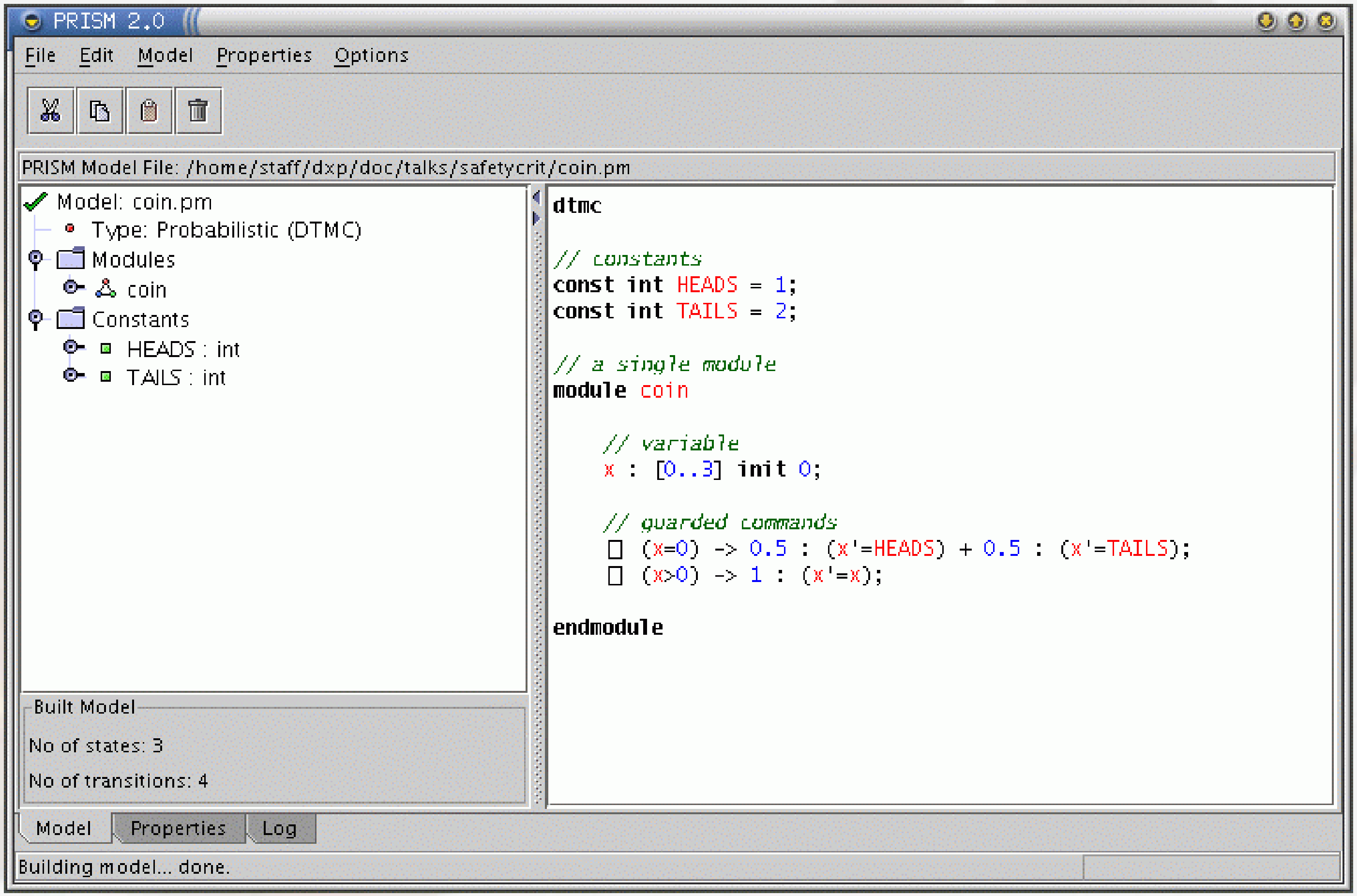
Adding costs/rewards

- Instantaneous rewards
  - state-based, e.g. "queue size", "concentration of reactant"
  - R=? [I=T], expected reward at time instant T?
  - R=? [S], expected long-run reward?
- Cumulative rewards
  - state- and transition-based, e.g. "time taken", "power consumed", "messages lost"
  - R=? [F A], expected reward to reach A?
  - R=? [C<=T], expected reward by time T?
  - R=? [S], expected long-run reward per unit time?

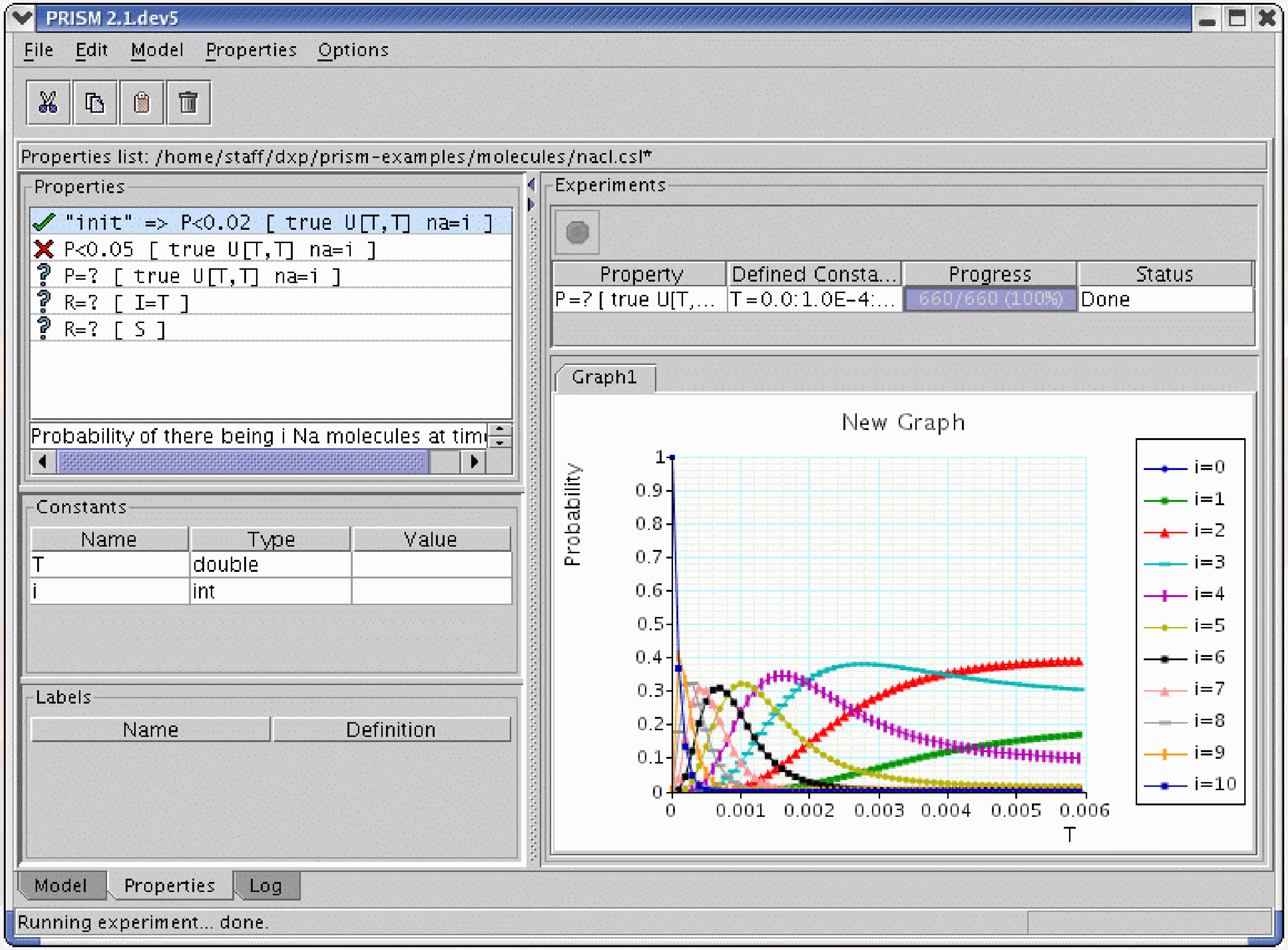
PRISM real-world case studies

- MDPs/DTMCs
  - Self-stabilising algorithms (based on Hermann and others)
  - Bluetooth device discovery [ISOLA'04]
  - Crowds anonymity protocol (by Shmatikov) [CSFW'02, JSC 2003]
  - Randomised consensus [CAV'01,FORTE'02]
  - Contract signing protocols (by Norman & Shmatikov) [FASEC'02]
  - NAND multiplexing for nano (with Shukla) [VLSI'04,TCAD 2005]
- CTMCs
  - Molecular reactions (based on Regev & Shapiro)
  - Eukaryotic cell cycle control (based on Lecca & Priami)
  - Dependability of embedded controller [INCOM'04]
  - Dynamic power management [HLDVT'02, FAC 2005]
- PTAs
  - IPv4 ZeroConf dynamic configuration [FORMATS'03]
  - Root contention in IEEE 1394 FireWire [FAC 2003, STTT 2004]
  - IEEE 802.11 (WiFi) Wireless LAN MAC protocol [PROBMIV'02]

Screenshot: Text editor

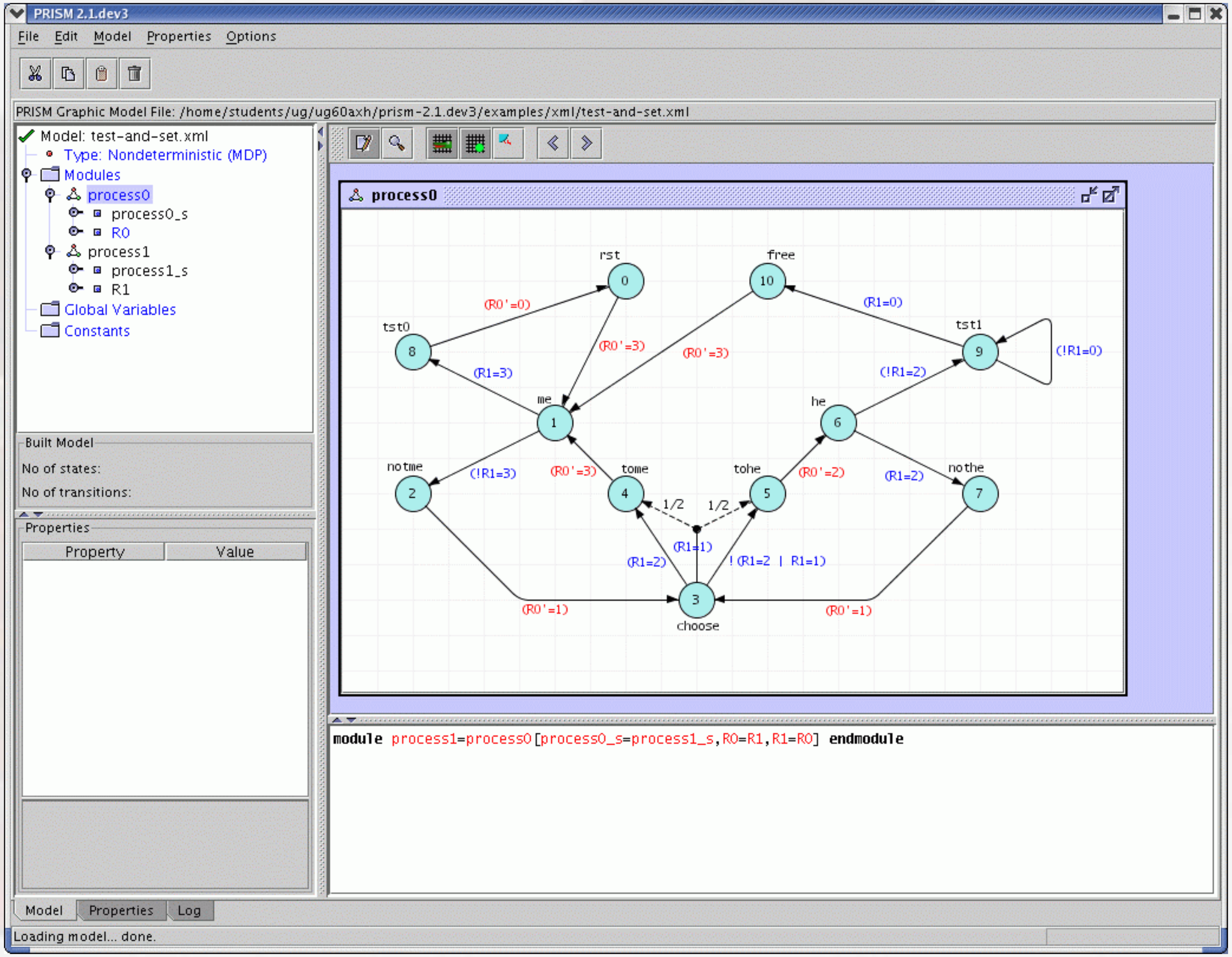


Screenshot: Graphs



Ongoing developments

- Graphical modelling language
- Simulator, sampling methods
- Parallel engine
- Grid engine



Case Study: Self-stabilization

- Self-stabilizing protocol for a network of processes
  - starts from possibly illegal start state
  - returns to a legal (stable) state
    - without any outside intervention
    - within some finite number of steps

- Network: synchronous or asynchronous ring of N processes
  - Illegal states: more than on process is privileged (has a token)
  - Stable states: exactly one process is privileged (has a token)
  - Properties
    - From any state, a stable state is reached with probability 1
    - Expected time to reach a stable state
    - Interested in worst-case time to reach stable state (unproven conjecture about Hermann's ring of McIver & Morgan)

Herman's self-stabilising protocol

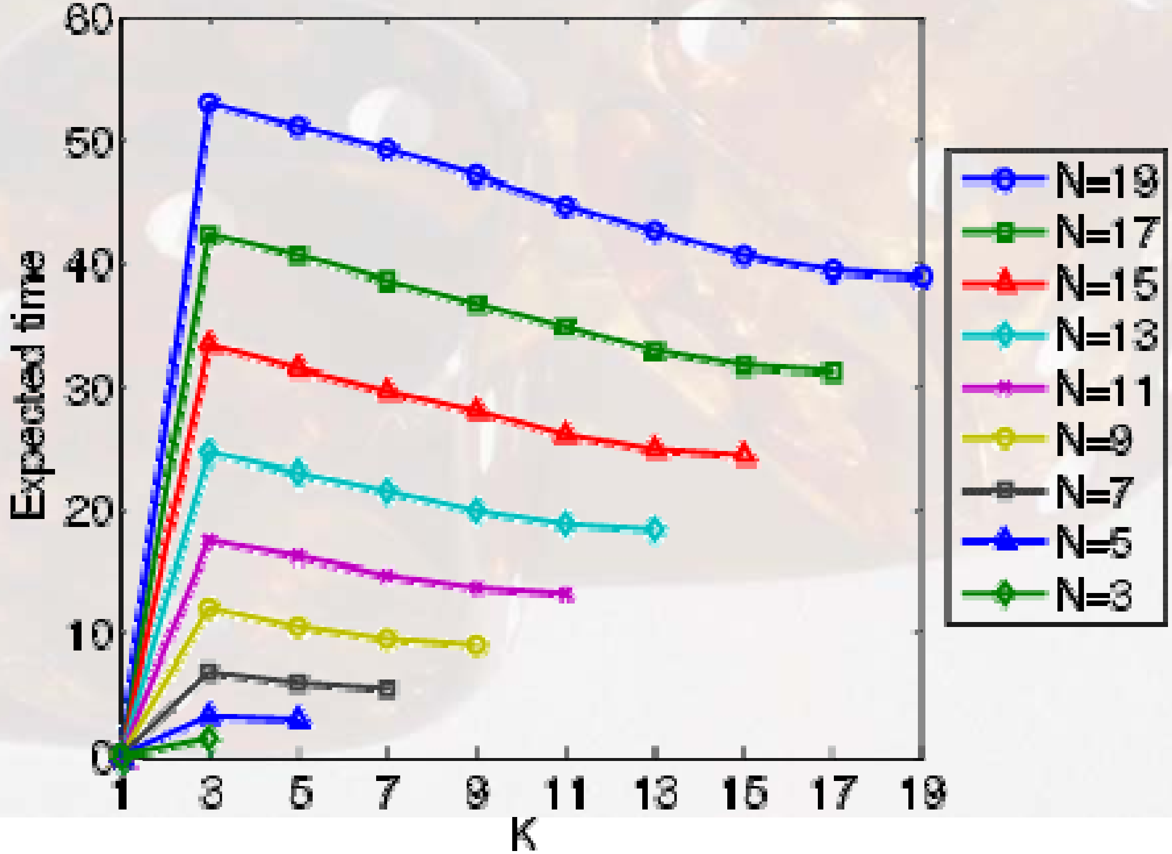
- Synchronous ring of N (N odd) processes (DTMC)
  - Each process has a local boolean variable x_i
  - Token in place i if $x_i=x_{i+1}$
  - Basic step of process i:
    - if $x_i = x_{i+1}$ make a uniform random choice as to the next value of x_i
    - otherwise set x_i to the current value of x_{i+1}
  - Allow to start in any state (MDP)
  - In the PRISM language:

```
module process1
    x1 : bool;
    [step]    x1=x2     -> 0.5 : x1'=0 + 0.5 : x1'=1;
    [step]  !(x1=x2)  -> x1'=x2;
endmodule

module process2  = process1 [x1=x2, x2=x3] endmodule
                 ⋮                             ⋮
module processN = process1 [x1=xN, x2=x1] endmodule
```

Results: Herman's protocol

- $P_{\geq 1}$ (◊stable): min probability of reaching a stable state is 1
- $E_{=?}$ (stable): max expected time (number of steps) to reach a stable state, assuming initially K tokens and N processes:



Israeli-Jalfon's self-stabilising protocol

- Asynchronous ring of N processes (MDP)
- Each process has a local boolean variable q_i
  - token in place i if q_i=true
  - process is active if and only if has a token
  - basic step of (active) process: uniform random choice as to whether to move the token to the left or right

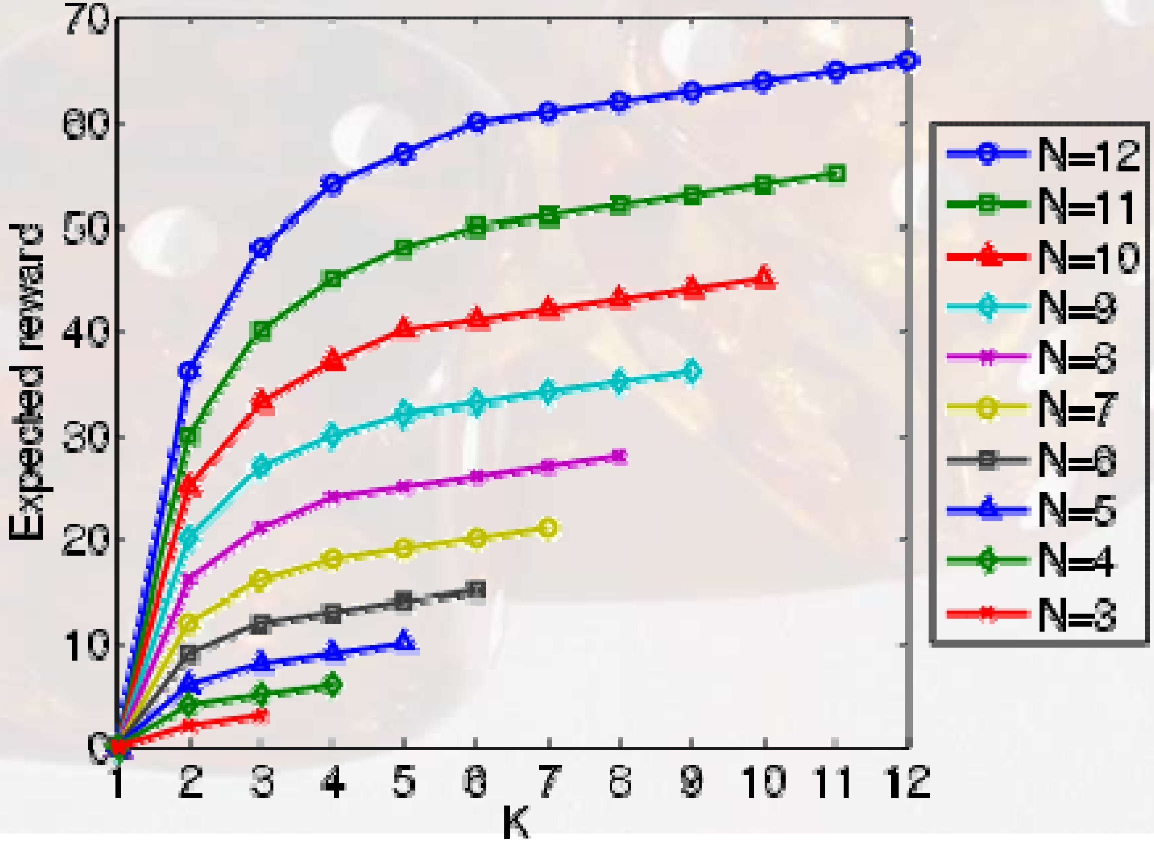
  - In the PRISM language:

```
global q1 : [0..1]; … global qN : [0..1];
module process1
    s1 : bool; // dummy variable
    [] (q1=1) -> 0.5 : (q1'=0) & (qN'=1) + 0.5 : (q1'=0) & (q2'=1);
endmodule

module process2  = process1 [s1=s2, q1=q2, q2=q3 , qN=q1] endmodule
                       ⋮                       ⋮
module processN = process1 [s1=sN, q1=qN, q2=q1 , qN=qN-1] endmodule
```

Results: Israeli-Jalfon's protocol

- P ¸ 1 (◊stable): min probability of reaching a stable state is 1
- E . ? (stable): max expected time (number of steps) to reach a stable state, assuming initially K tokens and N processes:



Case Study: Molecular Reactions

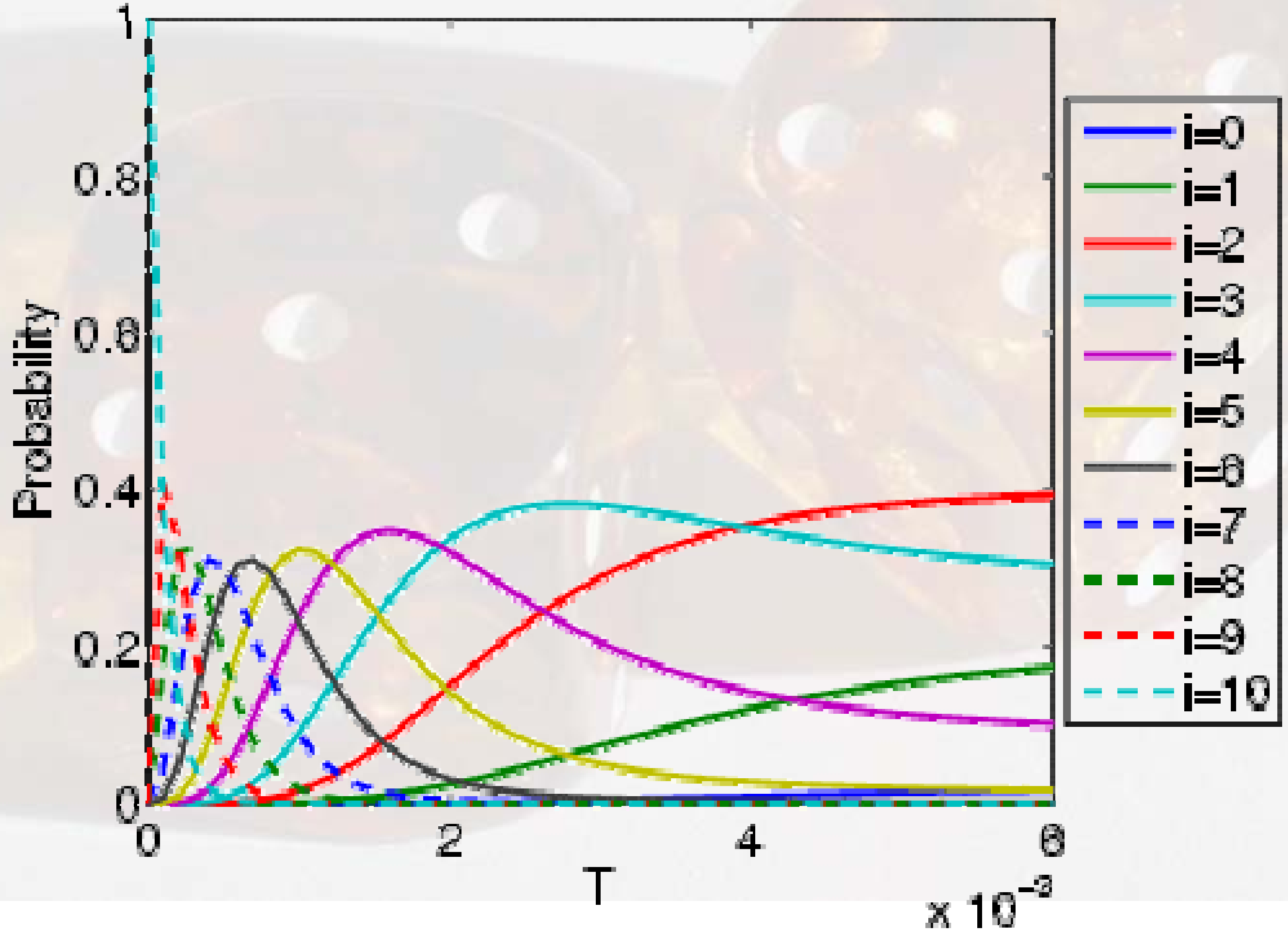
- Time until a reaction occurs is given by an exponential distribution [Gillespie 1977]
  - model reactions using continuous time Markov chains

- Rate of reaction determined by:
  - base rate (empirically determined constant)
  - concentration of reactants (number of each type of molecule that takes part in the reaction)

- This case study: **$Na + Cl \leftrightarrow Na^+ + Cl^-$**
  - forward base rate 100
  - backwards base rate 10
  - initially N1 Na molecules and N2 Cl molecules

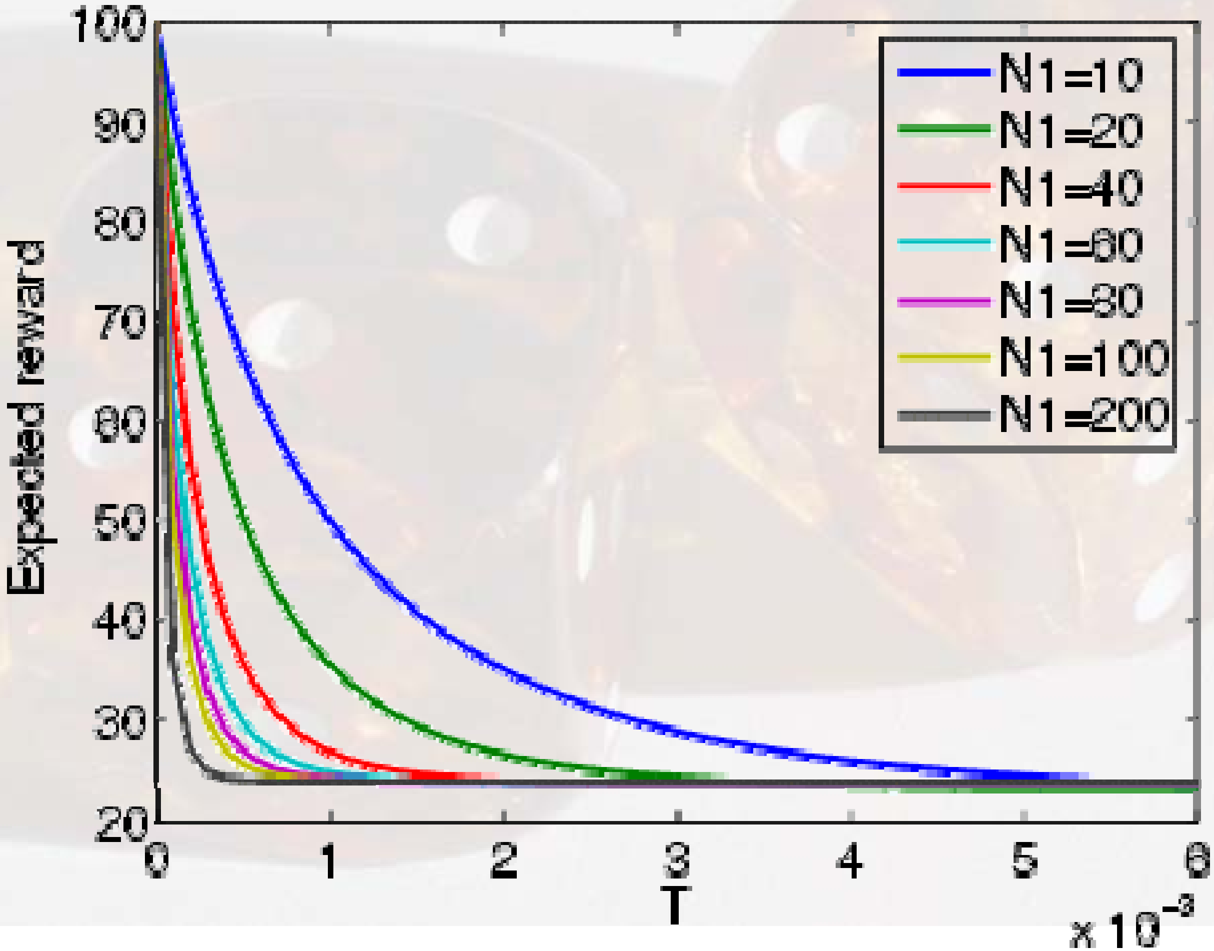
Results: Molecular Reactions

- $P_{=?}$ (true $U^{[T,T]}$ Na=i): probability of i Na molecules at time T



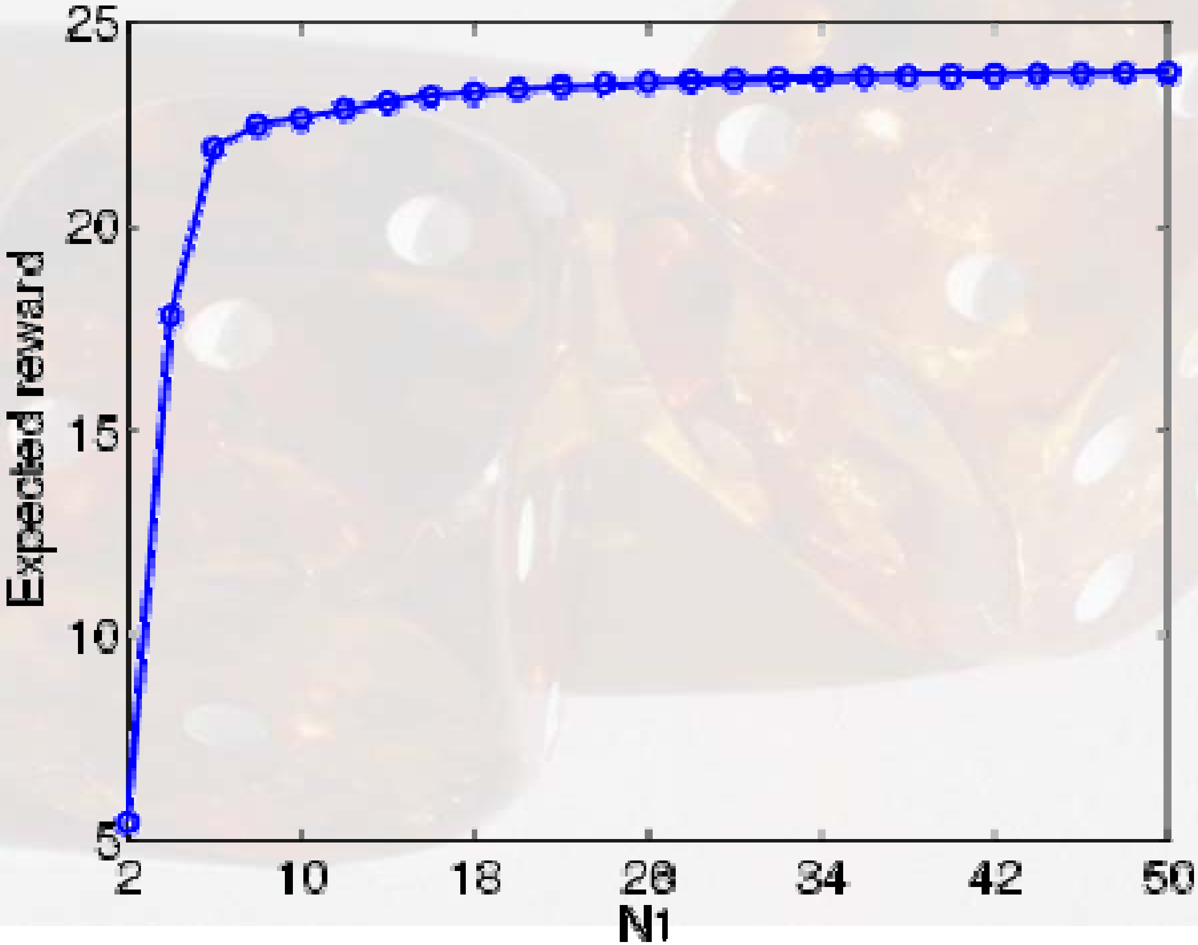
Results: Molecular Reactions

- $R_{=?}$ (I=T): expected percentage of Na molecules at time T



Results: Molecular Reactions

- $R_{=?}$ (S): expected percentage of Na molecules in the long run

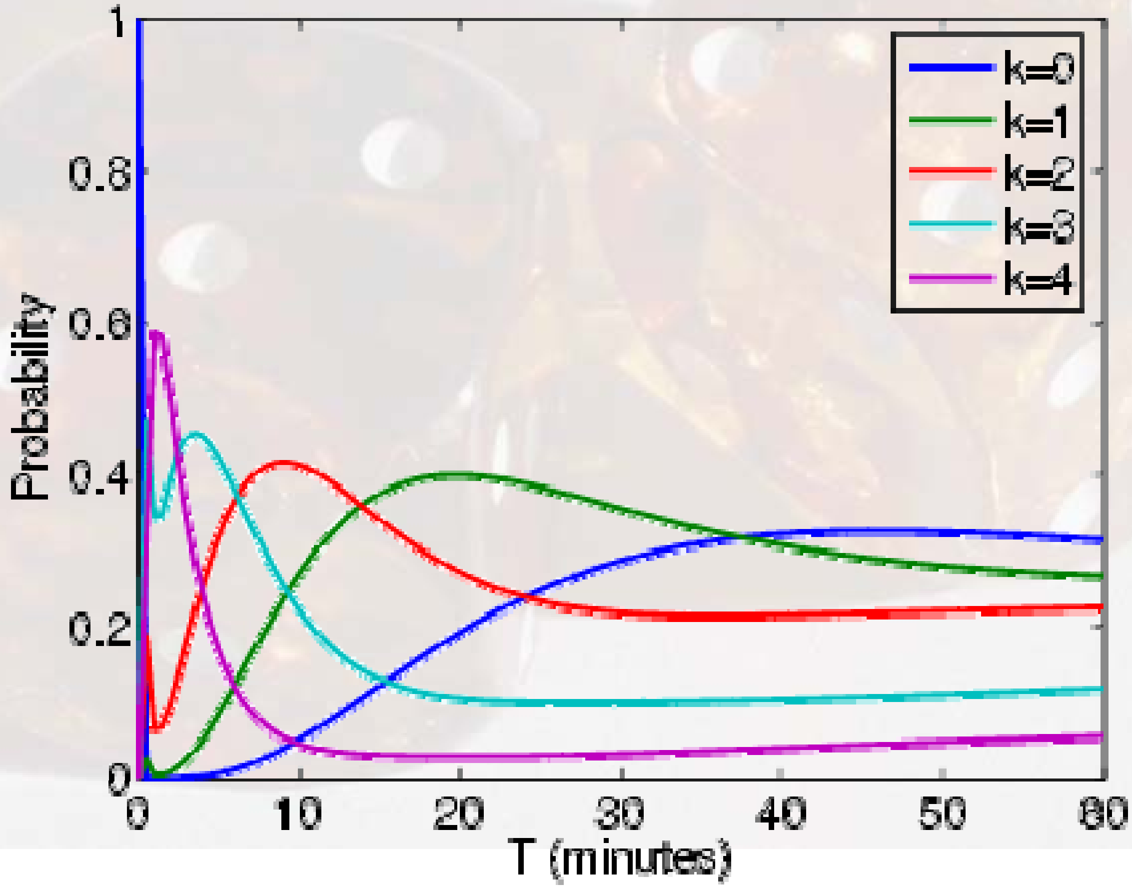


Case Study: Cell Cycle Control

- Eukaryotes
  - very common occurring class of single-celled or multi-celled organisms
- This case study: cell cycle control
- Based on earlier work work of Lecca and Priami
  - formal specification given in the **π-calculus**
  - simulation based approach (using BioSPI)
  - study the relative concentration of a number of types of proteins, partaking concurrently in several complex chemical reactions
- Construct PRISM model based on **π-calculus** specification
  - complements the simulation based approach

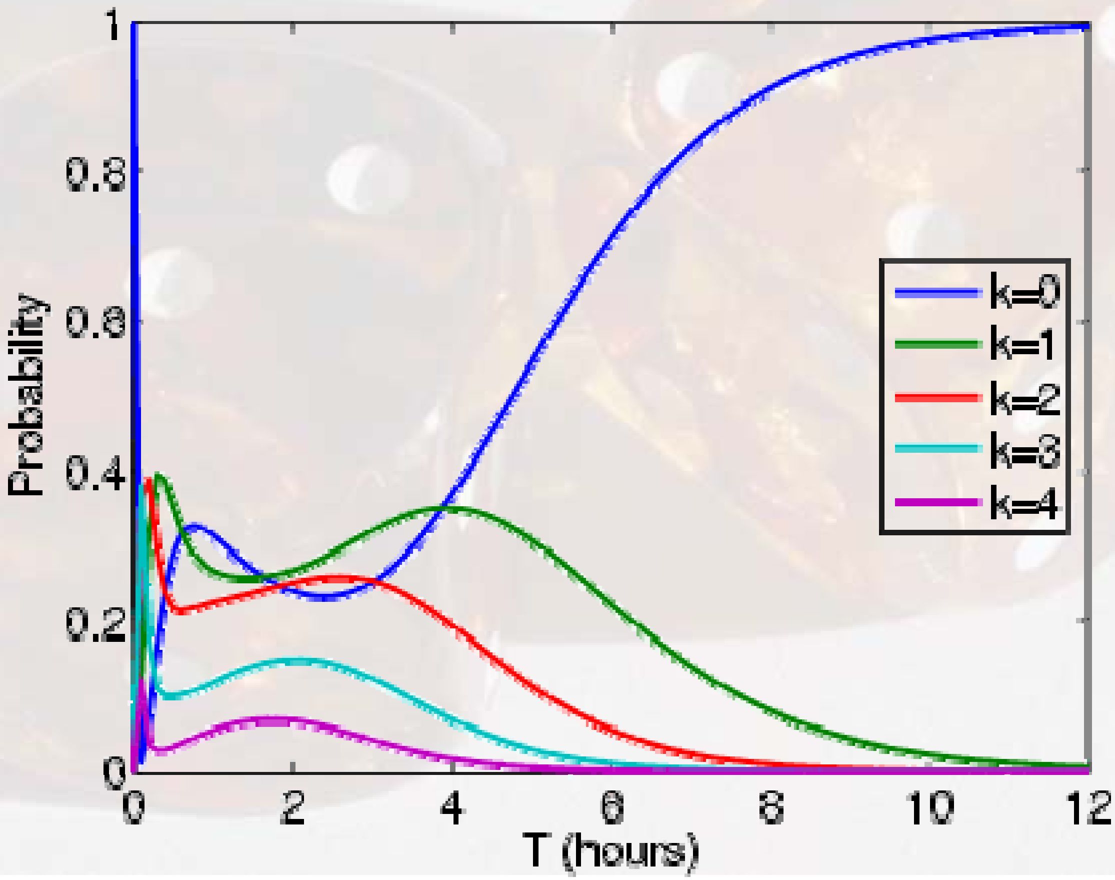
Results: Cell Cycle Control

- P$_{=?}$ (true U$^{[T,T]}$ cyclin=k): quantity of cyclin bound at time T equals k



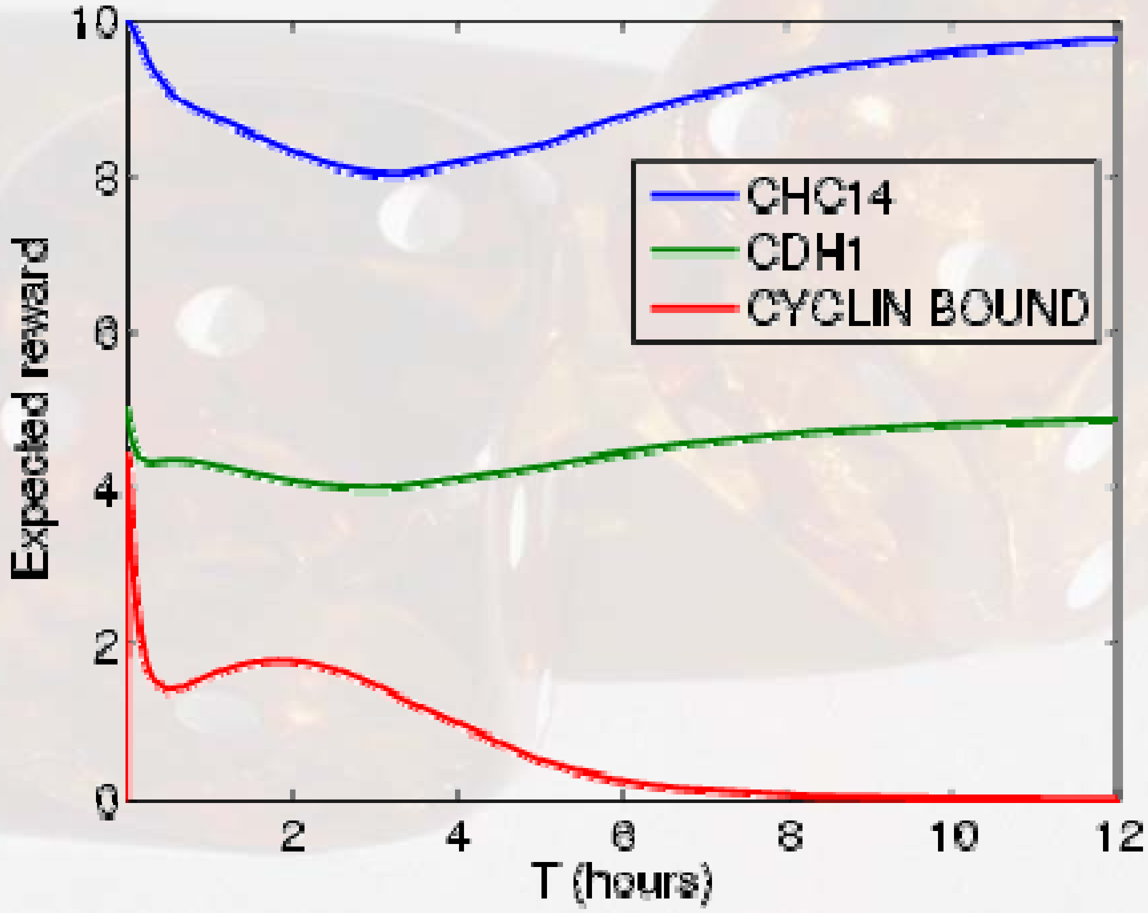
Results: Cell Cycle Control

- $P_{=?}$ (true $U^{[T,T]}$ cyclin=k): quantity of cyclin bound at time T equals k



Results: Cell Cycle Control

- $R_{=?}$ (I=T): expected quantities at time T



Case Study: Contract Signing

- Case study by Norman & Shmatikov [FASEC'02]

- Two parties want to agree on a contract
- Each will sign if the other will sign
  - Cannot trust other party in the protocol
  - There may be a trusted third party (judge), but it should only be used if something goes wrong

- Contract signing with pen and paper
  - Sit down and write signatures simultaneously
- Contract signing on the Internet
  - Challenge: how to exchange commitments on an asynchronous network?

Contract Signing

Partial secret exchange protocol of Even, Goldreich and Lempel (1985) for two parties (**A** and **B**)

- **A** (**B**) holds secrets $a_1,\dots,a_{2n}$ ($b_1,\dots,b_{2n}$)
  - Secret is a binary string of length l
  - Secrets partitioned into pairs: $\{(a_i, a_{n+i}) \mid i=1,\dots,n\}$ and $\{(b_i, b_{n+i}) \mid i=1,\dots,n\}$
  - **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 ½ otherwise receives **y**
  - **S** does not know which one **R** receives
  - if **S** cheats then **R** can detect this with probability ½

Contract Signing

(step 1)

for $i=1,\ldots,n$

  OT(A,B, a_i, a_{n+i})

  OT(B,A b_i, b_{n+i})

end

(step 2)

for $i=1,\ldots,l$ (l is the bit length of the secrets)

  for $j=1,\ldots,2n$

    A transmits bit i of secret a_j to B

  end

  for $j=1,\ldots,2n$

    B transmits bit i of secret b_j to A

  end

end

Results: Contract Signing

- Discovered a weakness in the protocol when party **B** is allowed to 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 at least one of ***A*** pairs, then, with probability 1, at this point:
    - **B** possesses a pair of ***A***'s secrets
    - ***A*** does not have complete knowledge of any pair of **B**'s secrets

- Protocol is therefore not fair under this attack:
  - **B** has a distinct advantage over ***A***

Results: Contract Signing

- 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 with probability ½ **B** (or **A**) gains this advantage
- We consider 3 possible alternate message sequence schemes…

Contract Signing: EGL2

```
(step1)
        …
(step2)
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
end
for i=1,…,l
        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
end
```

Contract Signing: EGL3

(step1)

...

(step2)

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

end

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

end

Contract Signing: EGL4

(step1)

…

(step2)

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

end

for i=1,…,l

A transmits bit i of secret a_{n+1} to B

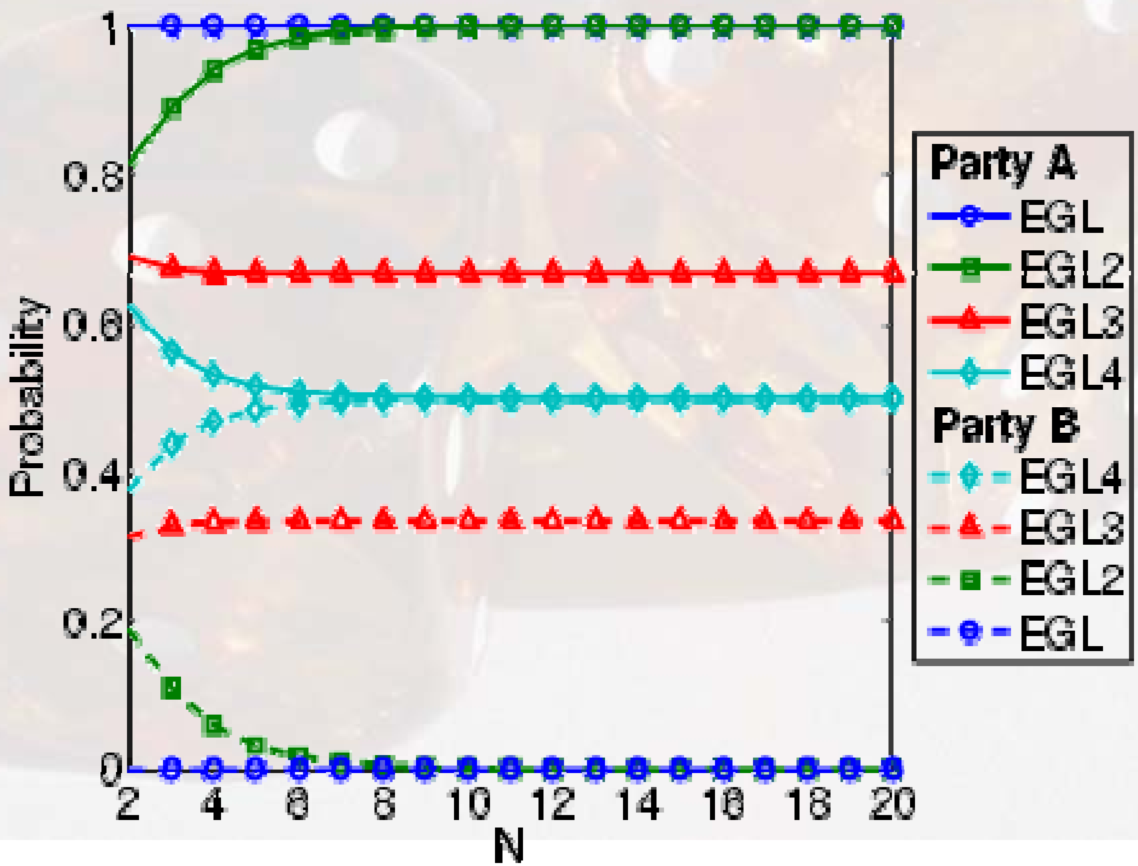
for j=n+1,…,2n B transmits bit i of secret b_j to A

for j=n+2,…,2n A transmits bit i of secret a_j to B

end

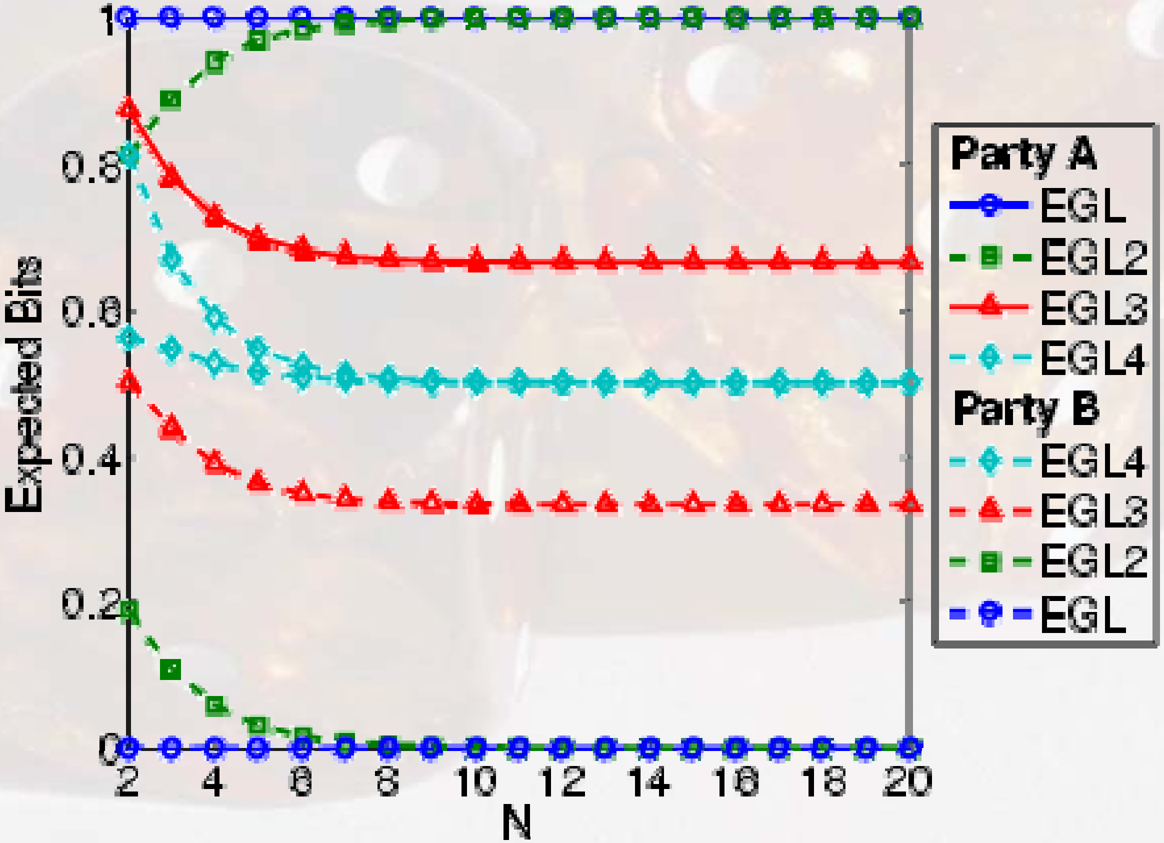
Results: Contract Signing

- Probability the other party gains knowledge first (the chance that the protocol is unfair)



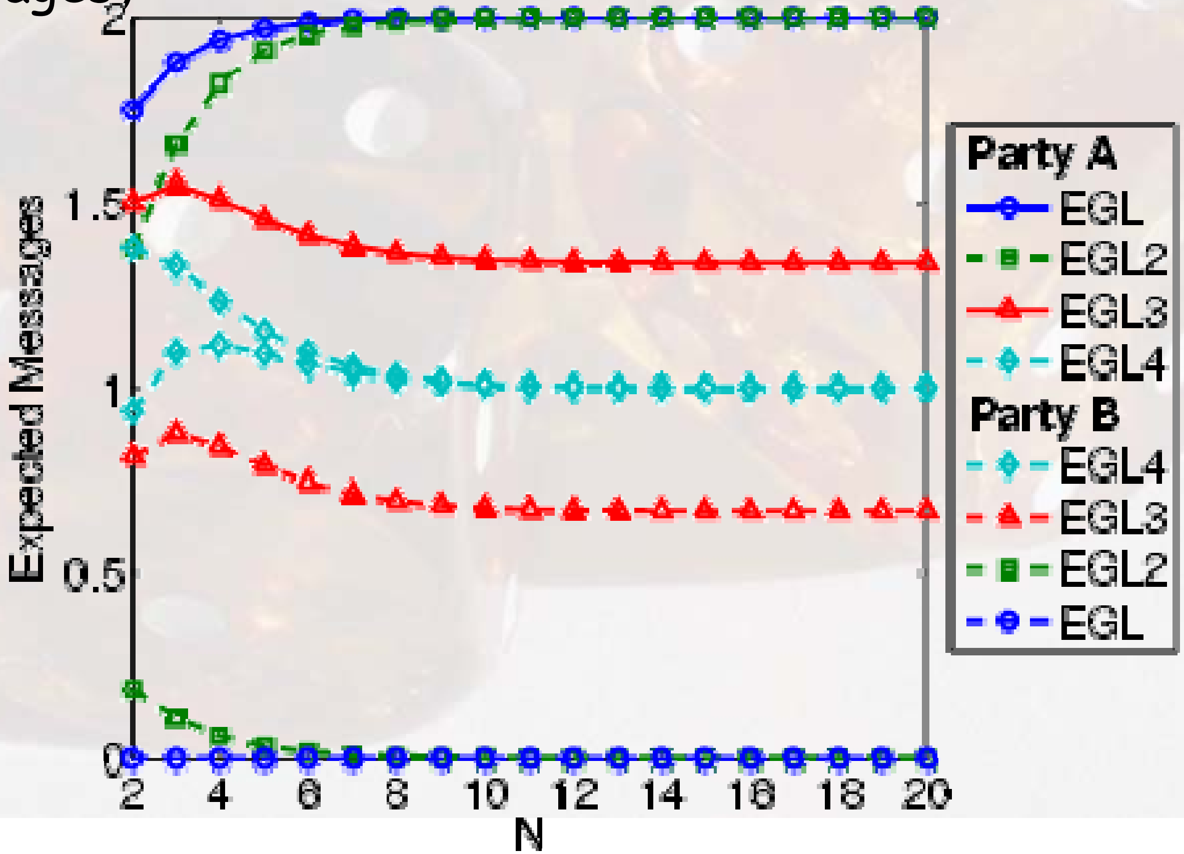
Results: Contract Signing

- Expected bits a party requires to know a pair once the other knows a pair (quantifies how unfair the protocol is)



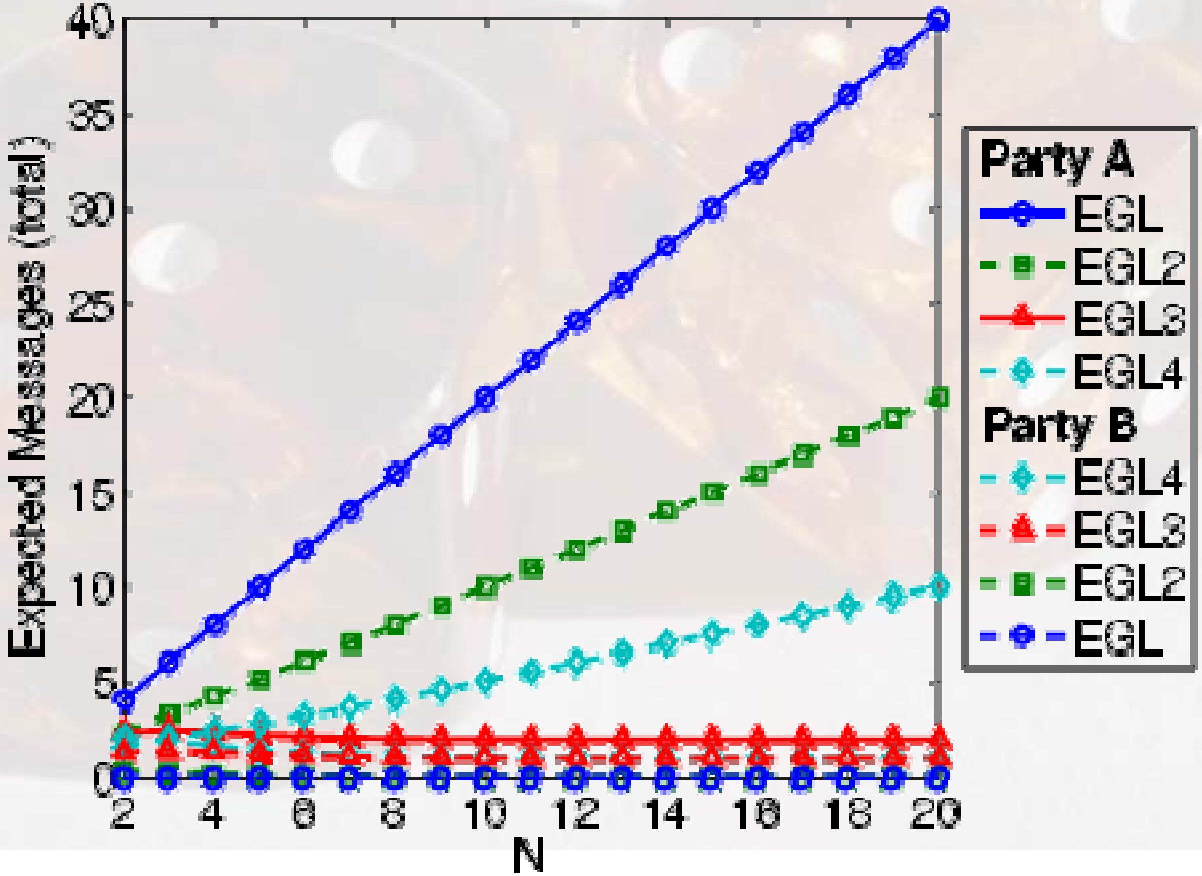
Results: Contract Signing

- Expected messages a party must receive to know a pair once the other knows a pair (measures the influence the other party has on the fairness, since it can try and delay these messages)



Results: Contract Signing

- Expected messages that need to be sent for a party to know a pair once the other party knows a pair (measures the duration of unfairness)



Results: Contract Signing

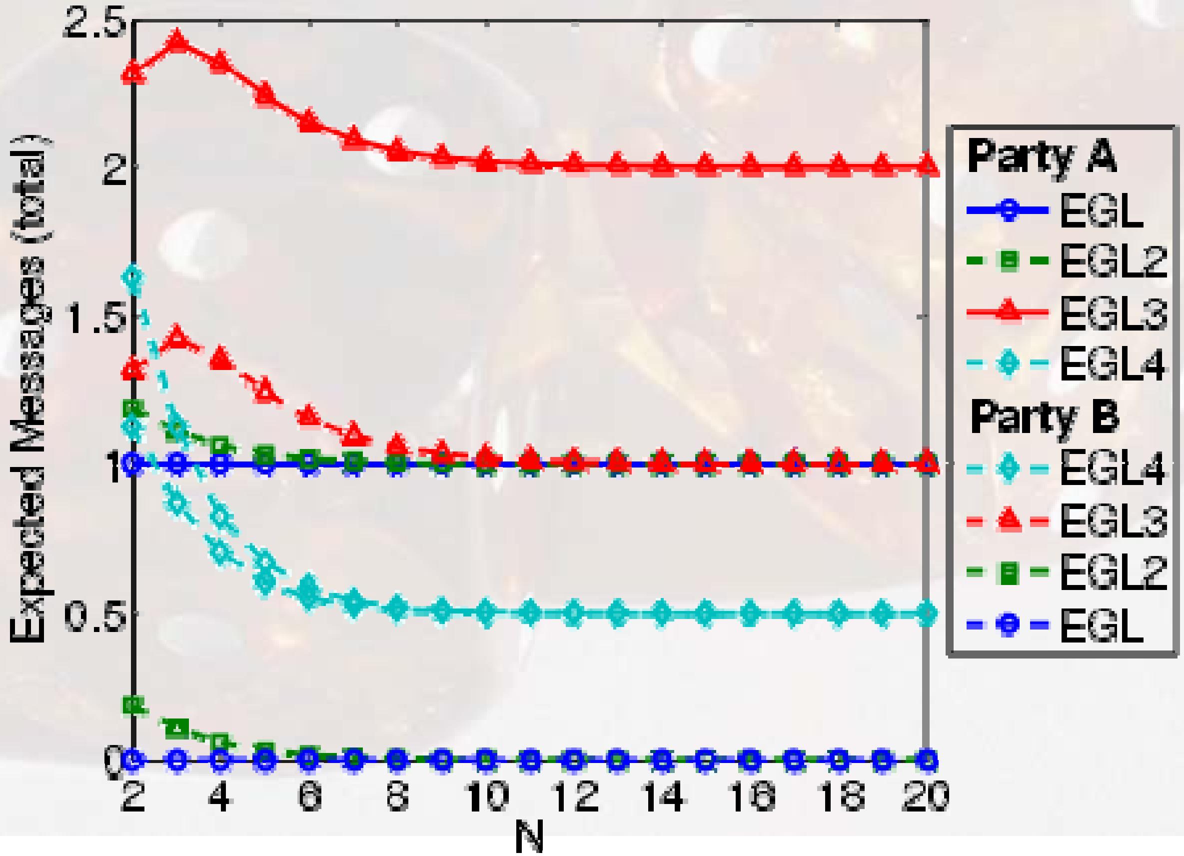
- Results show EGL4 is the 'fairest' protocol
- Except for duration of fairness measure:

  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**
  - and, in fact, as **n** increases, this measure:
    - increases for **B**
    - 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 as a single message

Results: Contract Signing

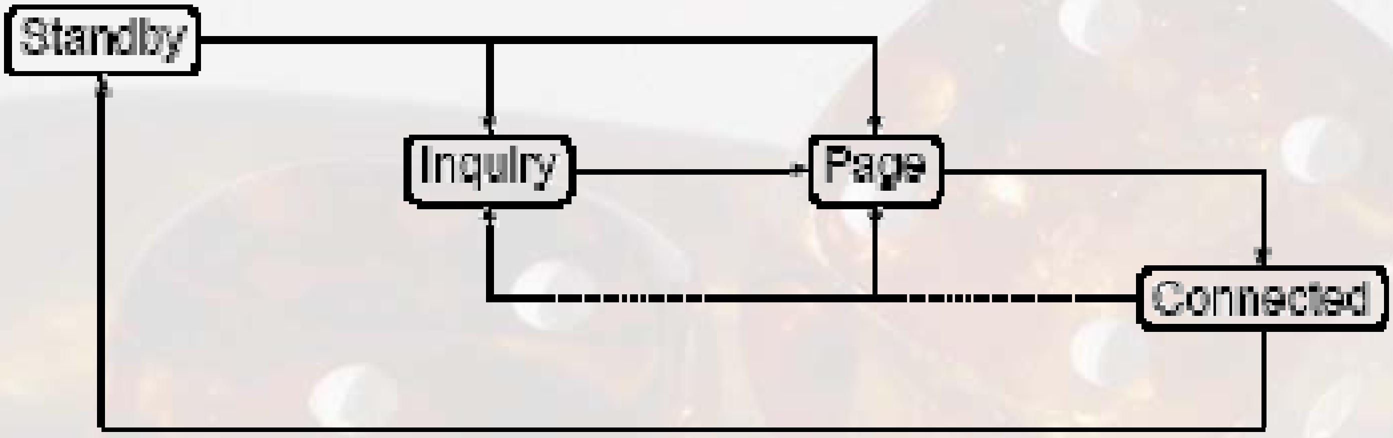
- Expected messages that need to be sent for a party to know a pair once the other party knows a pair (measures the duration of unfairness)



Case Study: Bluetooth Device Discovery

- Short-range low-power wireless protocol
  - Personal Area Networks (PANs)
  - Open standard, versions 1.1 and 1.2
  - Widely available in phones, PDAs, laptops, …
- Uses frequency hopping scheme
  - To avoid interference (uses unregulated 2.4GHz band)
  - Pseudo-random frequency selection over 32 of 79 frequencies
  - Inquirer hops faster
  - Must synchronise hopping frequencies
- Network formation
  - Piconets (1 master, up to 7 slaves)
  - Self-configuring: devices discover themselves
  - Master-slave roles

States of a Bluetooth device



- Master looks for device, slave listens for master
- Standby: default operational state
- Inquiry: device discovery
- Page: establishes connection
- Connected: device ready to communicate in a piconet

Frequency hopping

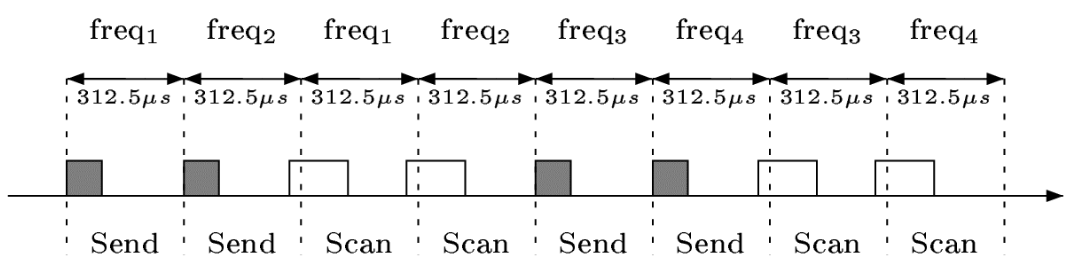


Fig. 1. Timing of the inquiring device's behaviour

- Clock CLK, 28 bit free-running, ticks every 312.5μs
- Inquiring device (master) broadcasts inquiry packets on two consecutive frequencies, then listens on the same two (plus margin)
- Potential slaves want to be discovered, scan for messages
- Frequency sequence determined by formula, dependent on bits of clock CLK (k defined on next slide):

$$freq = [CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}) \bmod 16] \bmod 32$$

Frequency hopping sequence

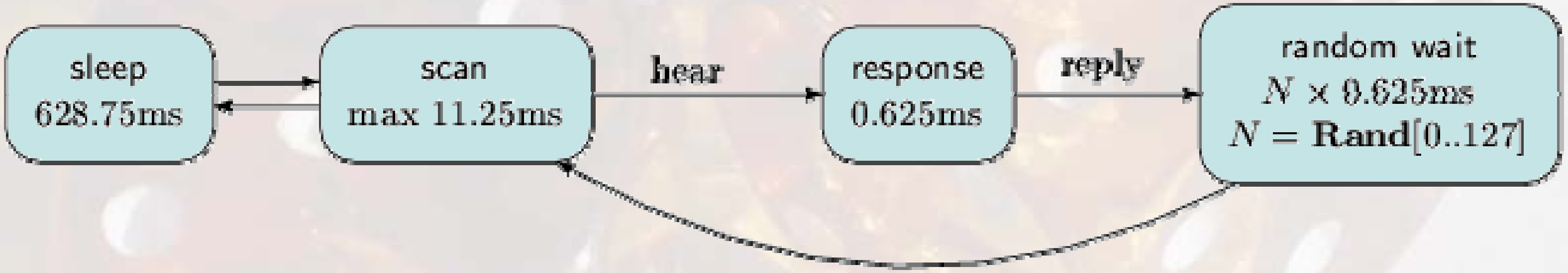
freq = [CLK_{16-12}+k+ ($CLK_{4-2,0}$- CLK_{16-12}) mod 16] mod 32

- Two trains (=lines)
- k is offset that determines which train
- Swaps between trains every 2.56 sec
- Each line repeated 128 times

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	16

Sending and receiving in Bluetooth

- Sender: broadcasts inquiry packets, sending according to the frequency hopping sequence, then listens, and repeats
- Receiver: follows the frequency hopping sequence, own clock



- Listens continuously on one frequency
- If hears message sent by the sender, then replies on the same frequency
- Random wait to avoid collision if two receivers hear on same frequency

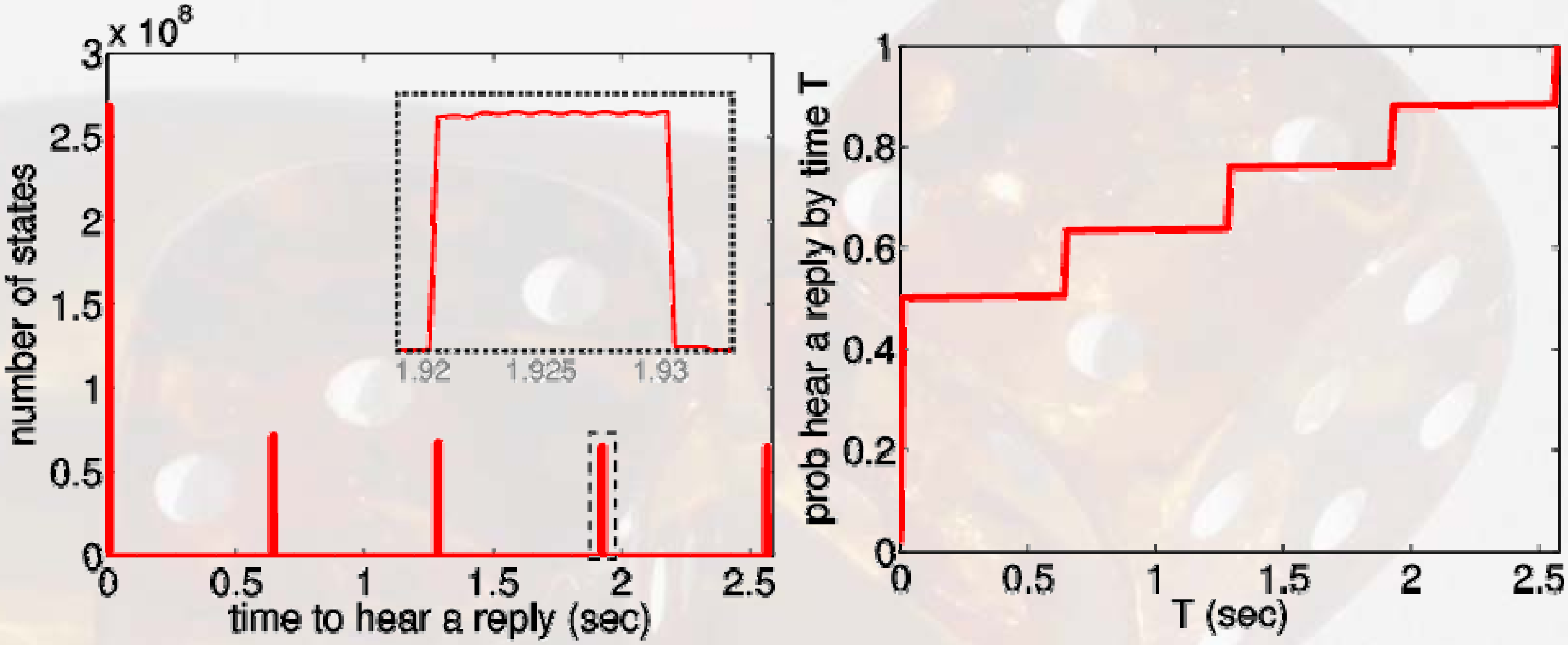
Bluetooth modelling

- Very complex interaction
  - Genuine randomness, probabilistic modelling essential
  - Devices make contact only if listen on the right frequency at the right time!
  - Sleep/scan periods unbreakable, much longer than listening
  - Cannot scale constants (approximate results)
  - Cannot omit subactivities, otherwise oversimplification
- Huge model, even for one sender and one receiver!
  - Initial configurations dependent on 28 bit clock
  - Cannot fix start state of receiver, clock value could be arbitrary
  - 17,179,869,184 possible initial states
- But is a realistic future ubiquitous computing scenario!

More about this Bluetooth model…

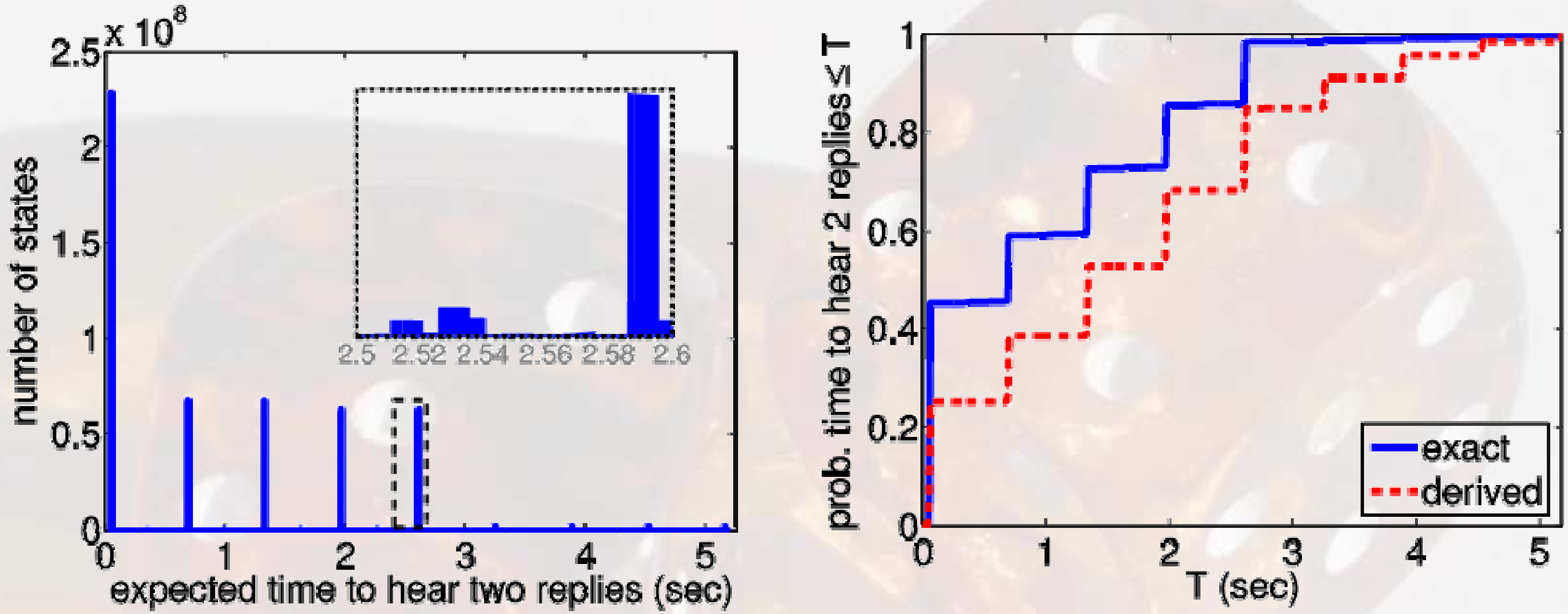
- Other approaches
  - network simulation tools (BlueHoc), obtain averaged results
  - analytical approaches, require simplifications to the model
  - easy to make incorrect probabilistic assumptions…
- Must optimise/reduce model
  - Assume negligible clock drift
  - Discrete time, obtain a DTMC
  - Divide into 32 separate cases
- Observations
  - Work with realistic constants, as in the standard
  - Analyse v1.2 and 1.1, confirm 1.1 slower
  - Show best/worst case values, can pinpoint scenarios which give rise to them
  - Also obtain power consumption analysis

Time to hear 1 reply



- Max time to hear is 2.5716sec, in 921,600 possible initial states, (Min 635μs)
- Cumulative: assume uniform distribution on states when receiver first starts to listen

Time to hear 2 replies



Huge probabilistic model, 17,179,869,184 possible initial states.
Max time is 5.177sec (16,565 slots), in 444 initial states.
Unlike simulation, model checking is exhaustive.
The exact curve is obtained by model checking.
Derived plot incorrectly assumes independence of events.

What we have learnt from practice

- Probabilistic model checking
  - Is capable of finding 'corner cases' and 'unusual trends'
  - Good for worst-case scenarios, for all initial states
  - Benefits from quantitative-style analysis for a range of parameters
  - Is limited by state space size
  - Useful for real-world protocol analysis, power management, performance, biological processes, …

- Simulation and sampling-based techniques
  - Limited by accuracy of the results, not state-space explosion
  - May need to rerun experiments for each possible start state, not always feasible
  - Statistical methods in conjunction with sampling help
  - Nested formulas may be difficult

PRISM successes so far

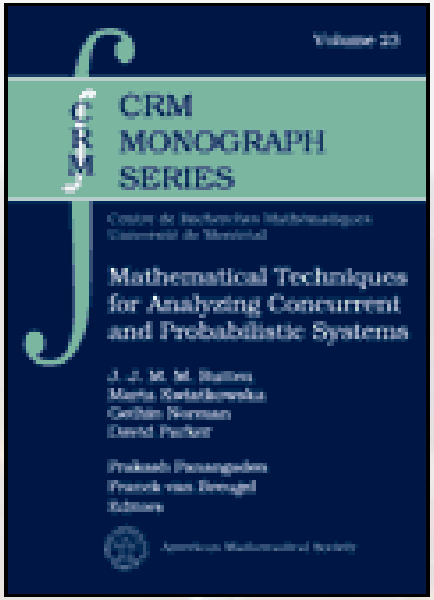
- Fully automatic, no expert knowledge needed for
  - Probabilistic reachability and temporal logic properties
  - Expected time/cost
- Tangible results!
  - 6 cases of "unusual behaviour" found, in over 30 case studies
  - Greater level of detail, has exposed obscure dependencies
- PRISM tool robust
  - Large, realistic models often possible
  - Choice of engines
- Essential to provide support for scalability
  - Abstraction, compositionality, …
  - Sampling-based methods, parallelisation, …

Challenges for future

- Exploiting structure
  - Abstraction, data/equivalence quotient, (de)compositionality…
  - Parametric probabilistic verification?
- Proof assistant for probabilistic verification?
- Efficient methods for continuous models
  - Continuous PTAs? Continuous time MDPs? LMPs?
- More expressive specifications
  - Probabilistic LTL/PCTL*/mu-calculus?
- Real software, not models!

- More applications
  - Quantum cryptographic protocols
  - Mobile ad hoc network protocols
  - Biological processes

For more information…



J. Rutten, M. Kwiatkowska, G. Norman and D. Parker
Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems
P. Panangaden and F. van Breugel (editors), CRM Monograph Series, vol. 23, AMS
March 2004

www.cs.bham.ac.uk/~dxp/prism/

- Case studies, statistics, group publications
- Download, version 2.1 (2000 downloads)
- Unix/Linux, Windows, Apple platforms
- Publications by others and courses that feature PRISM…

PRISM collaborators worldwide

Collaborators, contributors – thanks!

Rajeev Alur, Christel Baier, Roberto Barbuti, Muffy Calder, Stefano Cataudella, Stefano Cattani, Ed Clarke, Sadie Creese, Pedro D'Argenio, Conrado Daws, Luca de Alfaro, Marie Duflot, Amani El-Rayes, Wan Fokkink, Laurent Fribourg, Stephen Gilmore, Michael Goldsmith, Rajeesh Gupta, Vicky Hartonas-Garmhausen, Boudewijn Haverkort, Thomas Herault, Holger Hermanns, Ulrich Herzog, Andrew Hinton, Joe Hurd, Michael Huth, Jane Hillston, Jane Jayaputera, Bertrand Jeannet, Thomas Herault, Joost-Pieter Katoen, Matthias Kuntz, Kim Larsen, Richard Lassaigne, Andrea Maggiolo-Schettini, Annabelle McIver, Rashid Mehmood, Stephane Messika, Paolo Milazzo, Carroll Morgan, Gethin Norman, Colin O'Halloran, Antonio Pacheco, Prakash Panangaden, Dave Parker, Sylvain Peyronnet, Claudine Picaronny, Mark Ryan, Roberto Segala, Vitaly Shmatikov, Sandeep Shukla, Markus Siegle, Jeremy Sproston, Tran Manh Ha Tran, Angelo Troina, Moshe Vardi, Fuzhi Wang, Hakan Younes

EPSRC QinetiQ BRITISH COUNCIL THE UNIVERSITY OF BIRMINGHAM