Design and Analysis of DNA Strand Displacement Devices using Probabilistic Model Checking

Dave Parker
School of Computer Science, University of Birmingham

Joint work with:
Matthew Lakin, Luca Cardelli, Marta Kwiatkowska and Andrew Phillips

Centre for Systems Biology, Birmingham, June 2012
Overview

• **Quantitative verification**
  – probabilistic model checking and PRISM

• **Modelling and analysis of biological systems**
  – a discrete stochastic approach
  – probabilistic model checking: “in-silico” experiments

• **Two-domain DNA strand displacement**
  – gate correctness, reliability and performance
  – design optimisation: garbage collection
  – a larger example: approximate majority
  – see: [Lakin/Parker/…, Royal Society Interface, 2012]

• **Summary, challenges & directions**
Verification via model checking

**Model checking**: Automatic formal verification of correctness properties of computerised systems

- **System**
- **Finite-state model**
- **Temporal logic specification**
- **¬EF fail**
- **Model checker** e.g. SMV, Spin
- **Result**
  - ✓
  - ✗
- **Counter-example**
Probabilistic model checking

• Why and what?

• Why probability?
  – unreliability (e.g. component failures)
  – uncertainty (e.g. message losses/delays over wireless)
  – randomisation (e.g. in protocols such as Bluetooth, ZigBee)
  – stochasticity (e.g. biological/chemical reaction rates)

• Quantitative properties
  – reliability, performance, quality of service, ...
  – “the probability of an airbag failing to deploy within 0.02s”
  – “the expected power usage of a sensor network over 1 hour”
  – “the expected time for a cell signalling pathway to complete”
**Probabilistic model checking**: Automatic verification of quantitative properties of systems with stochastic behaviour.
• Construction and analysis of finite probabilistic models
  – e.g. Markov chains, Markov decision processes, …
  – specified in high-level modelling formalisms
  – exhaustive model exploration (all possible states/executions)

• Automated analysis of wide range of quantitative properties
  – properties specified using temporal logic
  – “exact” results obtained via numerical computation
  – linear equation systems, iterative methods, uniformisation, …
  – as opposed to, for example, Monte Carlo simulations
  – efficient techniques from verification + performance analysis
  – mature tool support available
The PRISM tool

• **PRISM: Probabilistic symbolic model checker**
  – developed at Birmingham/Oxford University, since 1999
  – free, open source software (GPL), runs on all major OSs

• **Support for:**
  – models: Markov chains, Markov decision processes, ...
  – properties: PCTL, CSL, LTL, PCTL*, costs/rewards, ...

• **Features:**
  – simple but flexible high-level modelling language
  – user interface: editors, simulator, experiments, graph plotting
  – multiple efficient model checking engines (e.g. symbolic)

• **Many import/export options, tool connections**
  – in: (Bio)PEPA, stochastic π-calculus, DSD, SBML, Petri nets, ...
  – out: Matlab, MRMC, INFAMY, PARAM, ...

• **See:** [http://www.prismmodelchecker.org/](http://www.prismmodelchecker.org/)
PRISM – Case studies

• Randomised communication protocols
  – Bluetooth, FireWire, Zeroconf, 802.11, Zigbee, gossiping, ...

• Randomised distributed algorithms
  – consensus, leader election, self-stabilisation, ...

• Security protocols/systems
  – pin cracking, anonymity, quantum crypto, contract signing, ...

• Planning & controller synthesis
  – robotics, dynamic power management, ...

• Performance & reliability
  – nanotechnology, cloud computing, manufacturing systems, ...

• Biological systems
  – cell signalling pathways, DNA computation, ...

• See: www.prismmodelchecker.org/casestudies
Overview

• Quantitative verification
  – probabilistic model checking and PRISM

• Modelling and analysis of biological systems
  – a discrete stochastic approach
  – probabilistic model checking: “in-silico” experiments

• Two-domain DNA strand displacement
  – gate correctness, reliability and performance
  – design optimisation: garbage collection
  – a larger example: approximate majority

• Summary, challenges & directions
Modelling biological systems

- **Aim:** model a mixture of interacting molecules
  - multiple molecular species, interacting through reactions
  - cell signalling pathway, gene regulatory network, ...
  - fixed volume (spatially uniform), pressure and temperature

- **Simple example:**
  - 3 species A, B and AB; 3 reactions:
  - reversible binding of A and B to form AB; degradation of A

\[
\begin{align*}
A + B & \xrightleftharpoons[k_2]{k_1} AB \\
A & \xrightarrow[k_3]{} 
\end{align*}
\]

- **Two approaches to modelling**
  - discrete, stochastic
  - continuous, deterministic
Modelling biological systems

• **Discrete, stochastic approach**
  – (integer) counts of number of each molecule: \( x = (x_A, x_B, x_{AB}) \)
  – inherently stochastic process [McQuarrie, Gillespie]
  – continuous–time Markov chain with states \( x \)
  – stochastic simulation, numerical soln., probabilistic model checking, …

• **Continuous, deterministic approach**
  – (real–valued) concentrations: \([A], [B], [AB]\)
  – solution of system of coupled ordinary differential equations
  – good approximation of \( E[x] \) for very large num.s of molecules
Discrete stochastic approach

- **Chemical master equation**
  - state vector \( \mathbf{x} = (x_A, x_B, x_{AB}) \)
  - probability \( P(\mathbf{x}, t) \) that at time \( t \) there will be \( x_Z \) of species \( Z \)

\[
\frac{\delta P(\mathbf{x}, t)}{\delta t} = \sum_{i=1}^{3} a_i(\mathbf{x} - \mathbf{v}_i)P(\mathbf{x} - \mathbf{v}_i, t) - a_i(\mathbf{x})P(\mathbf{x}, t)
\]

- stoichiometric vectors: \( \mathbf{v}_1 = (-1, -1, 1) \), \( \mathbf{v}_2 = (1, 1, -1) \), \( \mathbf{v}_3 = (-1, 0, 0) \)
- \( a_i(\mathbf{x}) \) are time-independent propensity functions
- mass-action: proportional to reactant combinations
  - e.g. \( a_1(x_A, x_B, x_{AB}) = k_1 \cdot x_A \cdot x_B \)

- **Stochastic process**: continuous-time Markov chain (CTMC)
  - transition rates (of exponential delays) derived from \( a_i \)
Continuous–time Markov chain (CTMC)

- **CTMC C = (S, s_i, R)**
  - states \( S \), initial state \( s_i \in S \)
  - rate matrix \( R : S \times S \rightarrow \mathbb{R}_{\geq 0} \)
  - \( R(s,s') \): rate of exponential delay before moving \( s \rightarrow s' \)
  - probability \( s \rightarrow s' \) triggered before time \( t = 1 - e^{-R(s,s') \cdot t} \)

- **Example: CTMC with:**
  - states \((x_A, x_B, x_{AB}) \in S = \{0,1,2\}^3\)
  - initial state \((2,2,0)\)

- **Rates for reactions**
  - \( r_1 \) (binding): rate = \( x_A \cdot x_B \cdot k_1 \)
  - \( r_2 \) (unbinding) rate = \( x_{AB} \cdot k_2 \)
  - \( r_3 \) (degradation): rate = \( x_A \cdot k_3 \)
Probabilistic model checking

Probabilistic model checking for systems biology...

Biological system

System model

CTMC

PRISM

Result

Quantitative results

Counter-example

System properties

Temporal logic

\[ P=? \left[ F^t \quad a>0 \right] \]
PRISM modelling language

- Simple, textual, state-based modelling language
  - for Markov chains (and other models)

- Language basics
  - networks formed from interacting modules
  - state of each module given by finite-ranging variables
  - behaviour of each module specified by guarded commands
  - interactions between modules through synchronisation
  - interactions are associated with state-dependent rates

\[
[r_1] \quad (a > 0) \quad \rightarrow \quad k_1 \cdot a \quad : \quad (a' = a - 1) \& (ab' = ab + 1);
\]

- action
- guard
- rate
- update
module A
    a : [0..N] init N;
    ab : [0..N] init 0;
    [r_1] a>0 → k_1*a : (a'=a-1) & (ab'=ab+1);
    [r_2] ab>0 → k_2*ab : (a'=a+1) & (ab'=ab-1);
    [r_3] a>0 → k_3*a : (a'=a-1);
endmodule

module B
    b : [0..N] init N;
    [r_1] b>0 → b : (b'=b-1);
    [r_2] b<N → b : (b'=b+1);
endmodule

Reactions $r_1/r_2$:

**A + B**  \[\xrightarrow{k_1} \xleftarrow{k_2} AB\]

Reaction $r_3$:

**A**  \[\xrightarrow{k_3} \]

Example ($r_1$):

(a,ab,b)  \[\xrightarrow{k_1 \cdot a \cdot b} \]

(a-1,ab+1,b-1)
Property specifications

- **Property specifications are based on temporal logic**
  - PRISM uses continuous stochastic logic (CSL) + extensions
  - also supports linear temporal logic (LTL)
  - flexible, compact, unambiguous definition
  - small subset of patterns/templates in common use
  - can express properties about the probability of occurrence of an event or the expected value of some cost/reward measure

- **CSL example:** $P_{>0.9} [ F^{\leq T} k_{pp}>0 ]$
  - “with probability greater than 0.9, at least some MAPK is activated within the first $T$ seconds”

- **Usually focus on “quantitative” CSL:** $P_{=} [ F^{\leq T} k_{pp}>0 ]$
  - “what is the probability that at least some MAPK is activated within the first $T$ seconds?”
  - typically compute/plot for a range of parameter values
Example (FGF)

- Probability that a signal is present at time $T$
  \[ P_{=?} \left[ F^T (\text{FRS2}_{GRB}>0 \& \text{relocFRS2}=0 \& \text{degFRS2}=0) \right] \]
More examples of (extended) CSL

- $P_{= \tau} \left[ F_{[t,t]} \ a=i \right]$ 
  - “the probability that there are exactly $i$ A after $t$ seconds”

- $P_{= \tau} \left[ F \ a=0 \right]$ 
  - “probability that all A proteins are eventually degraded”

- $S_{= \tau} \left[ c+d>M \right]$ 
  - “long-run probability that the total number of Cs and Ds activated is above $M$”

- $P_{= \tau} \left[ c=0 \ U>^{t} c>0 \ \{c=0\}^{\text{max}} \right]$ 
  - “highest probability of it taking more than $t$ seconds for C to become activated, from any state where there are none”

- $P_{= \tau} \left[ F \ c=N \right] / P_{= \tau} \left[ F \ c>0 \right]$ 
  - “the (conditional) probability that all C proteins are eventually activated, given that at least some of them are”

- $R_{t^{\text{“active_d”}}}=? \left[ l=t \right]$ 
  - “the expected number of activated D at time instant $t$”
Case studies

• Fibroblast Growth Factor (FGF) pathway
  – [Heath/Kwiatkowska/Norman/Parker/Tymchyshyn/Gaffney]
  – 12 species, 14 sets of reaction rules
  – model checking (PRISM) + simulation (stochastic π-calculus)
  – “in-silico” experiments: systematic removal of components
  – results validated by subsequent lab experiments

• RKIP–inhibited ERK pathway [Calder/Vyshemirsky/Gilbert/Orton]
  – model checking using PEPA and PRISM models
  – formal analysis highlighted errors in existing models
  – corrected models then validated against experimental data

• And more: Codon bias, Ribosome kinetics, Sorbitol dehydrogenase, T Cell Signalling Events, …
  – www.prismmodelchecker.org/casestudies/index.php#biology
Overview

• Quantitative verification
  – probabilistic model checking and PRISM

• Modelling and analysis of biological systems
  – a discrete stochastic approach
  – probabilistic model checking: “in-silico” experiments

• Two-domain DNA strand displacement
  – gate correctness, reliability and performance
  – design optimisation: garbage collection
  – a larger example: approximate majority

• Summary, challenges & directions
Two-Domain DNA Strand Displacement

- DNA computing with a restricted class of DNA strand displacement structures
  - double strands with nicks (interruptions) in the top strand
    - and two-domain single strands consisting of one (short) toehold domain and one recognition domain
  - "toehold exchange": branch migration of strand $<t^x>$ leading to displacement of strand $<x t^>$

- Used to construct transducers, fork/join gates
  - which can emulate Petri net transitions

Example: Transducer

- Transducer: converts input \(<t^x>\) into output \(<t^y>\)
Example: Transducer

- **Transducer: full reaction list**

  \[
  \begin{align*}
  &txttatata \quad \leftrightarrow \quad tx \\
  &txttatata \quad \leftrightarrow \quad ta \\
  &xtytatat \quad \leftrightarrow \quad at \\
  &xtytatat \quad \leftrightarrow \quad yt \\
  &xtytatat \quad \rightarrow \quad xt \\
  &txttatat \quad \rightarrow \quad ta \\
  &txttatat \quad \rightarrow \quad x \\
  &txttatat \quad \rightarrow \quad a \\
  \end{align*}
  \]

  Input \quad \longleftrightarrow \quad \text{unreactive structures (no exposed toeholds)} \quad \leftrightarrow \quad Output
DNA programming

- Challenge: correct, reliable designs; avoid interference

- [Cardelli’10] proposes a “nick algebra” to formalise the definition and behaviour of these two-domain DNA strands
  - syntax, algebraic equivalence relation, reduction rules

- Strict subset of DSD (DNA Strand Displacement) language
  - [Cardelli, Phillips, et al.]
  - accompanying software Visual DSD for analysis/simulation
  - now extended to include auto-generation of PRISM models

- Example:
  
  new t@0.0003,0.1126
def T(N, x, y) =
  ( N* <t^ a> | N* <y t^> | N* t^:[x t^]:[a t^]:[a] | N* [x]:[t^ y]:[t^ a]:t^ )
  ( T(1, x, y) | 1 * <t^ x> )
Formalising correctness…

- identify states where gate has terminated correctly: "all_done"
- (correct number of outputs, no reactive gates left)

Check:

- (i) any possible deadlock state that can be reached must satisfy "all_done"
- (ii) there is at least one path through the system that reaches a state satisfying "all_done"

In temporal logic (CTL):

- \( A \ [ G \ "deadlock" \Rightarrow "all_done" ] \)
- \( E \ [ F \ "all_done" ] \)

Verify using PRISM…

- for one transducer: both properties true
- for two transducers in series: (ii) is true, but (i) is false
Transducer flaw

- PRISM identifies a 5-step trace to the "bad" deadlock state
  - problem caused by "crosstalk" (interference) between DSD species from the two copies of the gates
  - previously found manually [Cardelli’10]
  - detection now fully automated

- Bug is easily fixed (and verified)

Counterexample:
(1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,1,1,0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,1,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,1,1,0,0,1,1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,1,1,0,0,1,1,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0)
(0,0,1,0,1,1,1,0,1,0,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0)
We can also use PRISM to study the kinetics of the pair of (faulty) transducers:

- $P = ? \ [ F^{[T,T]} "deadlock" ]$
- $P = ? \ [ F^{[T,T]} "deadlock" \ & \ !"all\_done" ]$
- $P = ? \ [ F^{[T,T]} "deadlock" \ & \ "all\_done" ]$

success/error equally likely

![Graph showing probability over time]

- Blue line: Terminate
- Green line: Error
- Red line: Success

Probability

$T \times 10^4$
Transducers: Reliability

• Even without fixing the flaw in the transducer design...
  – we can improve reliability by using larger numbers of copies

• Plot: Expected number of reactive gates in the final state
  – for N copies of the faulty transducer pair
Transducers: Performance

- We analyse the performance of the (corrected) transducers
  - circuit composed of chain of K transducers
  - Seelig/Soloveichik showed execution time linear in depth

- Analysed for DSD model in PRISM:
  - $R_{\{\text{time}\}=? \ [ F \ "\text{all\_done}\" \ ]}$

![Graph showing expected time versus K with a linear relationship]
Catalysts in DSD

- Slightly more complex DSD gate design
  - extension of the transducer gate design
- Chemical reaction $X \rightarrow Z$ catalysed by $3^{rd}$ species $Y$
  - i.e. $X + Y \rightarrow Y + Z$
- Design decision:
  - can/should we implement garbage collection (GC)?
  - i.e. tidying up of intermediate species into inert structures
  - omitting GC makes design simpler and cheaper
  - but is it still correct?
    and what about efficiency?
- PRISM analysis:
  - both designs correct
  - GC speeds up gate execution significantly
  - due to extra reactions

![Graph](image)
Approximate Majority

- **Approximate majority population protocol** [Angluin et al.]
  - two populations X, Y and an auxiliary species B
  - aim is to converge to a consensus: either X or Y
  - should converge to population with initial majority

- **Reactions:**

\[
\begin{align*}
  X + Y & \xrightarrow{k_1} Y + B \\
  B + X & \xrightarrow{k_3} X + X \\
  Y + X & \xrightarrow{k_2} X + B \\
  B + Y & \xrightarrow{k_4} Y + Y
\end{align*}
\]

- We implement the approximate majority protocol in DSD
  - using the catalyst reactions shown earlier
  - and then analyse its correctness
Approximate majority: Simulation

- **Typical simulation run:**
  - in this instance, the protocol chooses $Y$
Approximate majority: Analysis

- Plot probability of choosing X for varying initial X/Y
  - 0.5 for equal initX and initY
  - rapidly approaches 1 as majority increases
Approximate majority: Analysis

- [Angluin et al.] prove correct consensus obtained with high probability if the initX-initY margin is above $\omega(\sqrt{N \log N})$
  - re-plot same data against (relative) initX-initY margin
  - for various total initial population sizes N (=4,…,10)
  - note increasingly clear threshold for larger N
Model checking DNA: Limitations

• **Key challenge (as always): state space explosion**
  - CTMCs solved for this work up to approx. 2m states

• **Already using various methods to reduce state space:**
  - careful gate design to reduce number of asynchronous steps
  - highest level of abstraction for reactions in DSD tool
  - for approximate majority, fuels modelled as “constant species”

• **Some positive results:**
  - bugs found in small systems, which also exist in bigger ones
  - we illustrated useful design trade-offs with small populations
  - earlier work (FGF): successful expt. validation for small sizes

• **On the other hand:**
  - transducer bug only arises for a transducer pair, not when studied in isolation; can we explore all possible interfaces?
  - how can we formally relate results obtained from smaller models to larger ones?
Overview

• Quantitative verification
  – probabilistic model checking and PRISM

• Modelling and analysis of biological systems
  – a discrete stochastic approach
  – probabilistic model checking: “in-silico” experiments

• Two-domain DNA strand displacement
  – gate correctness, reliability and performance
  – design optimisation: garbage collection
  – a larger example: approximate majority

• Summary, challenges & directions
**Summary**

- **Probabilistic model checking**
  - automatic, exhaustive construction of probabilistic models
  - analysis of formally specified quantitative properties
  - efficient techniques, tools available

- **Probabilistic model checking for systems biology**
  - discrete, stochastic model: chemical master equation
  - solution of continuous–time Markov chains
  - quantitative properties expressed in temporal logic

- **DNA strand displacement**
  - two–domain DSD designs analysed with Visual DSD, PRISM
  - correctness, reliability, performance, design decisions
Challenges and Directions

**Challenges**
- scalability, infinite-state systems
- correct level of abstraction for formal languages?
- appropriate (and testable) model checking queries?
- closer integration of model checking tools, engines

**Directions**
- model abstractions (and automatic construction of)
- infinite state systems: truncation for time-bounded properties
- model reduction techniques: bisimulation, symmetry, ...
- approximate/statistical model checking (simulation-based)
- stochastic hybrid systems: discrete + continuous populations
- compositional probabilistic model checking