



Modelling and Optimisation of a DNA Stack Nano-Device Using Probabilistic Model Checking

Bowen Li ✉ 

Interdisciplinary Computing and Complex bioSystems (ICOS) Research Group,
School of Computing, Newcastle University, Newcastle upon Tyne, UK

Neil Mackenzie ✉

Interdisciplinary Computing and Complex bioSystems (ICOS) Research Group,
School of Computing, Newcastle University, Newcastle upon Tyne, UK

Ben Shirt-Ediss ✉ 

Interdisciplinary Computing and Complex bioSystems (ICOS) Research Group,
School of Computing, Newcastle University, Newcastle upon Tyne, UK

Natalio Krasnogor ✉ 

Interdisciplinary Computing and Complex bioSystems (ICOS) Research Group,
School of Computing, Newcastle University, Newcastle upon Tyne, UK

Paolo Zuliani ✉ 

Interdisciplinary Computing and Complex bioSystems (ICOS) Research Group,
School of Computing, Newcastle University, Newcastle upon Tyne, UK

Abstract

A DNA stack nano-device is a bio-computing system that can read and write molecular signals based on DNA-DNA hybridisation and strand displacement. *In vitro* implementation of the DNA stack faces a number of challenges affecting the performance of the system. In this work, we apply probabilistic model checking to analyse and optimise the DNA stack system. We develop a model framework based on *continuous-time Markov chains* to quantitatively describe the system behaviour. We use the PRISM probabilistic model checker to answer two important questions: 1) What is the minimum required incubation time to store a signal? And 2) How can we maximise the yield of the system? The results suggest that the incubation time can be reduced from 30 minutes to 5-15 minutes depending on the stack operation stage. In addition, the optimised model shows a 40% increase in the target stack yield.

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases probabilistic model checking, CTMC, DNA computing, DNA stack

Digital Object Identifier 10.4230/LIPIcs.DNA.2022.5

Supplementary Material *Software (Source Code)*: <https://github.com/shell1bw/mcSTACK>
archived at `swh:1:dir:7b4a4cbb0e4f6ab425aedd6066c2a221f7dffefe`

Funding This work was partly funded by the EPSRC's project "Synthetic Portabolomics: Leading the way at the crossroads of the Digital and the Bio Economies" (EP/N031962/1). Krasnogor was supported by the Royal Academy of Engineering under the Chairs in Emerging Technologies scheme.

Acknowledgements The authors would like to thank Dr. Harold Fellermann for helpful advice.

1 Introduction

DNA computing is an emerging field that aims to use DNA, biochemistry, and molecular biology to construct information-processing devices. Since its first appearance where DNA was used for encoding the directed Hamiltonian path problem [1], the approach has been applied to design a wide range of biocomputing applications such as molecular computational circuits [7, 22], DNA storage technologies [9, 21], and synthetic controllers [8]. In [20], we



© Bowen Li, Neil Mackenzie, Ben Shirt-Ediss, Natalio Krasnogor, and Paolo Zuliani;
licensed under Creative Commons License CC-BY 4.0

28th International Conference on DNA Computing and Molecular Programming (DNA 28).

Editors: Thomas E. Ouldridge and Shelley F. J. Wickham; Article No. 5; pp. 5:1–5:22

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

presented the design and *in vitro* implementation of a DNA stack nano-device that can read and write molecular signals in a last-in first-out way. The stack data structure is implemented as a linear chain of partially complementary DNA strands which are used to represent both data and read/write operations. The operations are achieved via DNA-DNA hybridisation and strand displacement of complementary toehold domains. Although experimental results show a generally successful implementation, the system still faces a number of challenges affecting the performance of the resulting stack. That is in particular due to the presence of unwanted interactions among the biochemical species that implement the system.

To improve the performance of the DNA stack system, we propose the use of probabilistic model checking [4, 17] in the stack design process. Probabilistic model checking is a formal verification technique for modelling and analysing stochastic (probabilistic) systems. A benefit of the technique is the ability to exhaustively explore finite-state probabilistic models, typically Markov chains or variants. Since all possible behaviours of the system are analysed, the result is more reliable (and faster in some cases) than stochastic simulation where one simulation run follows a single possible trajectory of the system. Although probabilistic model checking was originally applied to software verification, the technique has been successfully used in biological settings as well [6, 14, 15, 19]. By quantitatively evaluating properties such as “*what is the probability for the system to reach steady-state within 10 minutes*”, we are able to predict the behaviour and optimise the design of a complex biochemical system.

In this work, we develop a probabilistic model framework based on *continuous-time Markov chains* (CTMCs) for describing the DNA stack system presented in [20]. We address two important challenges of the system: how to minimise the incubation time, and how to maximise the yield of the target stack. A set of properties based on the Continuous Stochastic Logic (CSL) are defined to quantitatively and rigorously analyse the dynamic behaviour of the DNA stack model. By model checking these properties we optimise the experimental protocol for constructing the DNA stack *in vitro*. The probabilistic model checker PRISM [18] is used for the model development and analysis.

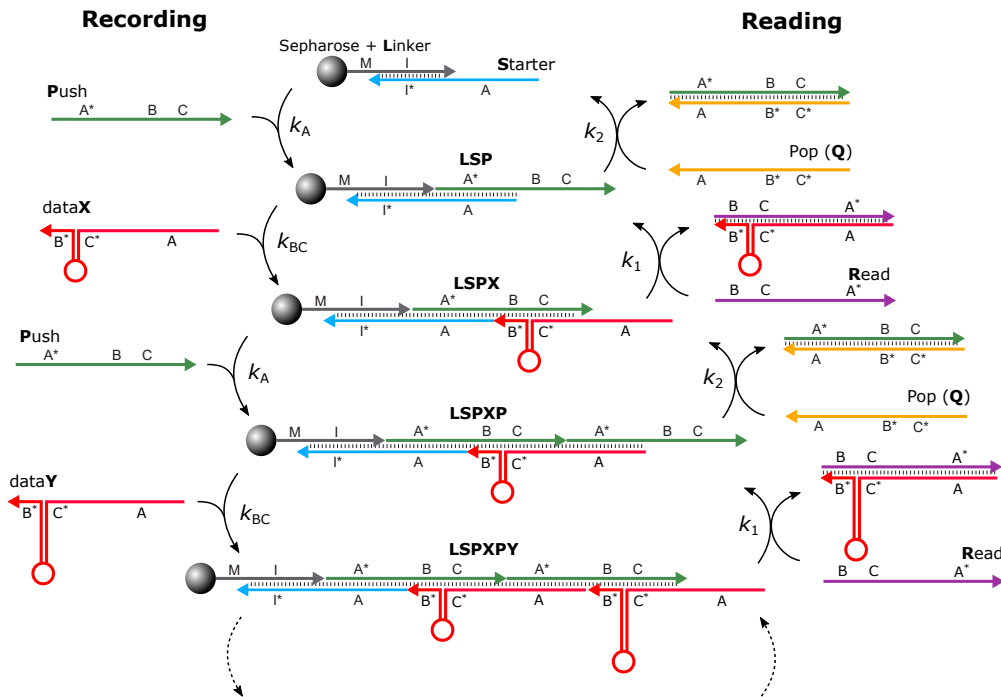
The rest of paper is organised as follows: Section 2 briefly introduces the concept of the DNA stack system, its experimental implementation, and CTMC-based probabilistic model checking. Section 3 describes the computational model of the DNA stack system, while Section 4 presents several validation and model checking results. Section 5 discusses experimental validation of the model checking results and concludes the paper.

2 Background

2.1 DNA stack system

We give here a brief overview of the DNA stack system – more details can be found in [12, 16, 20]. A computational stack is an abstract data type that serves as a linear collection of data elements, with two main operations: *push* (adds an element to the *head* of collection) and *pop* (removes from the *head* of the collection and returns the most recently added element). Because of this LIFO (Last In, First Out) feature, stacks are much used for enforcing sequential access to data.

A DNA stack is a *molecular* implementation of the stack data structure where the data elements that are stored, as well as the operations needed to operate the stack, are engineered via a set of single stranded DNA strands, also called “DNA data operators”; DNA data operators use both sequence and (partial) secondary structures information to achieve the functionality by mimicking their computational counterpart. The construction of a DNA stack structure is achieved via DNA-DNA hybridization and strand displacement of complementary



■ **Figure 1** Schematic of the DNA stack nano-device.

toehold domains. In a DNA stack implementation, the push operation prepares the stack for receiving an element, while pop undoes that. To record an element in the DNA stack, one performs a push operation followed by a data recording operation. To reverse this, a specific read operation is applied that removes the last element added, followed by a pop operation which undoes push and returns the stack to its initial state.

Figure 1 gives a schematic representation of a DNA stack system that performs two push and two pop operations for the data elements *dataX* (X) and *dataY* (Y). Six distinct DNA data operators were developed with their unique toehold domains which can be fully complementary (A and A*, B and B*, etc.). In addition to the toehold domains, each type of DNA data operator has a unique hairpin motif for the data storage. These hairpins do not participate in hybridisation or branch migration; they simply represent 0s and 1s (although, potentially, they could each store or encode more complex data).

The process to record an element starts from an empty stack, which is represented by hybridised *linker* and *start* strands (LS) with the linker attaching to streptavidin beads. The first *push* DNA data operator (P) is added to irreversibly hybridise with the empty stack via the exposed toehold domain A. The stack is now in its data state (LSP), and the corresponding reaction for this operation can be written as $LS + P \xrightarrow{k_A} LSP$. The LSP species now has a single open toehold region BC that can accept *dataX* (or *dataY*) DNA data operator via the reaction: $LSP + X \xrightarrow{k_{BC}} LSPX$. The process can be repeated by adding any number of *push* and then *data* DNA data operator. The reading process proceeds in a reversed way. Starting from a constructed stack, e.g., LSPXPY, the *read* DNA data operator peels the last recorded *dataY* DNA data operator off the stack by hybridisation at the exposed domain A and then three-way branch migration with domains BC. This operation forms the LSPXP stack and a *read-dataX* (RX) double stranded helix which does

not expose any single stranded domain and will not participate in further DNA interactions – the corresponding reaction is $LSPXPY + R \xrightarrow{k_1} LSPXP + RX$. Similarly, the *pop* DNA data operator (Q) peels off the exposed *push* DNA data operator, thus making the stack ready for another read (or write). The reaction also produces a double stranded helix *push-pop* (PQ): $LSPXP + Q \xrightarrow{k_2} LSPX + PQ$.

2.2 Experimental methods

The experimental protocols for the DNA stack are detailed in [20]. Briefly, 400 μL of Sepharose beads (Cytiva) are incubated and washed to remove the storage ethanol. Then 300nM of *linker* is added to the Sepharose beads and incubated at 25°C for 30 minutes. After the incubation is completed the species is washed (using a solvent of filtered 1X Tris EDTA(TE)) and the sample is centrifuged for 5s at max RPM. The supernatant is then carefully removed. The process is repeated for each sequential DNA data operator in the order *push*, *data* DNA data operator and so on, until the *releaser* is added to remove the constructed stack from the Sepharose beads. The stack is then analyzed via polyacrylamide gel electrophoresis.

The wash event performed prior to each DNA data operator addition aims to eliminate supernatants such as RX, PQ, P, X, *etc.* It is worth to notice that due to nonspecific binding to beads, the supernatant cannot be perfectly removed by the washing. Therefore, some of these residues can trigger side reactions to form undesired species in the next stack operation, which may harm the target stack production. For example, in the recording process of adding the second *push* DNA data operator, the stack LSPXP can also hybridise with residual *dataX* DNA data operator added in the previous step. Together with the new *push* DNA data operator P, different types of undesired polymers, such as LSPXPX, LSPXPXP, PXPX and others, can be formed in the chemistry.

2.3 Probabilistic model checking

Probabilistic model checking [4, 17] is a powerful technique for analysing systems that exhibit stochastic behaviour. Unlike classical model checking [10] which can only provide qualitative answers, i.e., the system either satisfies or violates a given property, probabilistic model checking can reason *quantitatively* about the probability that the property is true (or false). To probabilistically model check a system, two inputs are required: a probabilistic system model and the specification of a temporal property about the system behaviour. In this section, we introduce the DNA stack system model, and the language utilised for specifying the temporal properties.

2.3.1 Level-based CTMC

Continuous-time Markov chains (CTMCs) are a well-known stochastic model utilised for describing and analysing reaction networks [13]. Traditionally, the CTMC state is a population vector giving the number of molecules of each species. This representation, however, is infeasible for many real-life biological systems, which usually have large molecular numbers. In such cases, the size of the underlying CTMCs become huge: the so-called *state-space explosion* problem. Level-based CTMCs were introduced to model systems in a more abstract and efficient way [2, 6]. In a level-based CTMC, each state is characterised by molecular *concentrations*, discretised into a number of levels. In this paper we use a similar idea, but instead of concentrations levels we utilise *molecular count* levels for a more explicit state-space representation.

► **Definition 1** (Level-based CTMC). *A level-based continuous time Markov chain is a tuple (N, S, S_0, R) where N is the molecular count level number, S is a finite set of states; $S_0 \subseteq S$ is the set of initial states; and $R : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is the transition rate matrix. Moreover, each state $s \in S$ is a tuple $s = (n_1, n_2, \dots, n_k)$ for a fixed $k \in \mathbb{N}$ (number of species), where $n_i : 0 \leq n_i \leq N$ is the molecular count level for the i -th species.*

Thus, given a maximum molecular count M for all species, each level n_i represents the molecular count intervals $[0, \frac{M}{N}), [\frac{M}{N}, 2 \cdot \frac{M}{N}), \dots, [(N-1) \cdot \frac{M}{N}, N \cdot \frac{M}{N}]$ (level-based CTMCs are useful when $M \gg N$, hence by taking N to be a power of 10 prevents any rounding issues). Level-based CTMCs can be directly derived from stochastic reaction models – for deterministic models one can convert concentrations and rate constants as usual, or directly use concentration levels [2, 6]. Consider a simple reaction $A + B \xrightarrow{k} C$ with the stochastic rate constant k , and a possible state $s = (n_A, n_B, n_C)$ of species' molecular count levels. The transition rate is calculated as $\frac{n_A \cdot n_B \cdot R \cdot k}{R}$ where $R = \frac{M}{N}$.

2.3.2 Continuous Stochastic Logic

Continuous Stochastic Logic (CSL) [3, 5] is a formal notation to express probabilistic temporal properties of CTMCs and other dynamical systems. Formulae in CSL can be classified into state and path formulae:

$$\begin{aligned} \text{State formula } \Phi &::= \text{true} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid P_{\bowtie p}[\varphi] \mid S_{\bowtie p}[\varphi] \\ \text{Path formula } \varphi &::= X^I\Phi \mid \Phi U^I\Phi \end{aligned}$$

where $a \in AP$ is an atomic proposition over the states of a CTMC, $p \in [0, 1]$ is a probability threshold, $\bowtie \in \{\leq, <, \geq, >\}$, and I is an interval of $\mathbb{R}_{\geq 0}$. State formulae are evaluated over states of a CTMC, while path formulae are evaluated over paths (or executions/traces) of a CTMC. Formulae $P_{\bowtie p}[\varphi]$ and $S_{\bowtie p}[\varphi]$ are transient-state and steady-state CSL properties, respectively. In this work, we only consider transient-state properties since our DNA stack has a finite reaction time. A CTMC state s satisfies $P_{\bowtie p}[\varphi]$ if the probability of all the paths starting from s and satisfying φ , satisfies the bound $\bowtie p$. There are two types of operators in a path formula: the *next* operator where $X^I\Phi$ is true if Φ is satisfied in the next state of the path and at a time point $t \in I$, and the *until* operator where $\Phi_1 U^I\Phi_2$ is true if Φ_2 is satisfied at some time point within interval I , and that Φ_1 is true up until that point. Additional path operators can be derived as:

- the *eventually* operator F (future) where $F^I\Phi := \text{true } U^I\Phi$, and
- the *always* operator G (globally) where $G^I\Phi := \neg F^I\neg\Phi$

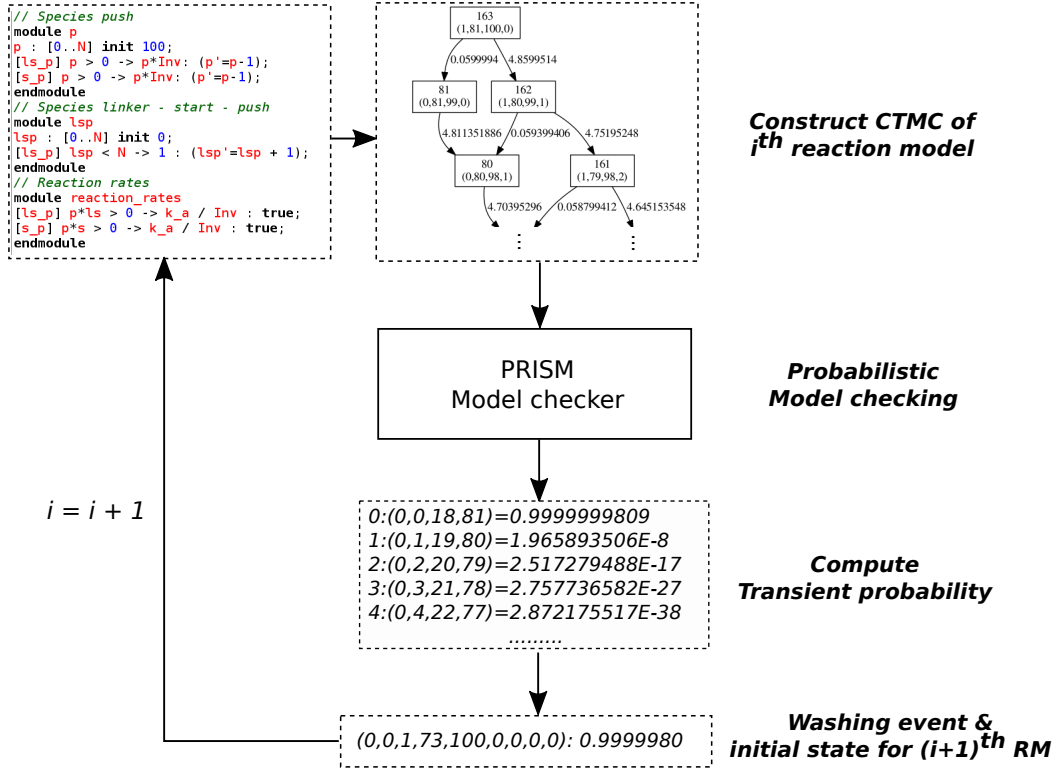
Intuitively, $F^I\Phi$ expresses that Φ is eventually satisfied at some time point within I , whereas a path satisfies $G^I\Phi$ if Φ is true at every time point in I . (See [5] for the formal CSL semantics.)

3 Modelling DNA stack system

3.1 Workflow overview

We have extended the PRISM model checker by implementing an iterative workflow¹ to model the process of the DNA stack system (Figure 2). PRISM offers a high-level modelling language for describing system behaviour. Following the styles defined in [11, 6], we developed

¹ Source code available at <https://github.com/shelllbw/mcSTACK>.



■ **Figure 2** CTMC-based workflow for modelling the DNA stack system.

a set of level-based PRISM reaction models (RMs) covering the possible reactions in each DNA operation step. Starting from the first RM which describes the reactions after first adding *push* DNA data operator, the workflow is as follow:

- **Construct CTMC based on i^{th} RM:** if $i = 0$, the initial state for generating the first CTMC is defined as an assumed state of the DNA chemistry such that some initial steps have been performed (see Appendix for details). Otherwise, the initial states are obtained from the result of the previous step. Here we use the *explicit* PRISM engine for building CTMCs as our model has a potentially large state space, but only a small fraction of which is actually reachable. The engine takes less than 10s to build a CTMC with 10^6 states on a 3.4 GHz Intel Core i5 processor with 12GB memory – much faster than the *MTBDD* engine which takes hours.
- **Compute transient probability:** the transient probability at the end of the incubation time is calculated for each CTMC state. Each state represents a possible molecular level that the system can reach at the end of incubation.
- **Perform wash event:** the transient states and their probabilities are modified in order to model the wash event. Washing is assumed to be an instantaneous event. All the state variables corresponding to the bead-bound species (e.g., *LSP*, *LSPX*) have their value N changed to $(1 - \mu)N$, where $\mu \in [0, 1]$ is the constant fraction of beads lost on each wash. All the state variables corresponding to the non bead-bound supernatant species (e.g., *X*, *RX*, *PQ*) have their value N' changed to $\phi N' B_m$, which models the survival from wash

event due to non-specifically bind to beads. Here $\phi \in [0, 1]$ is the constant fraction of supernatant solution transferred through the wash cycle to the next reaction stage, and $B_m = (1 - \mu)^n$ is the normalised mass of beads remaining where n is the index of the current wash. Moreover, states with the same value after the modification are merged and their probabilities accumulated.

- **Generate initial state of $(i + 1)^{\text{th}}$ RM:** in addition to the wash event, transient states are further manipulated for constructing the next RM. This includes extending the states with new variables (i.e., new species formed in the next reaction stage) with the corresponding state values (i.e., initial level of newly added DNA data operators). To reduce the state space size, a cut-off probability of 10^{-5} is introduced so that only states with relatively high probability are considered as initial states in the next step.

3.2 Reaction models

The major challenge of model checking DNA stack systems is the *state space explosion* problem. A stack system with several operations can potentially generate huge numbers of reachable states after only a few iterations. It is therefore important not only to consider the reaction models that we developed are able to describe system behaviour, but also to make sure a reasonable state space size can be maintained for the model analysis. To alleviate the problem, we have applied several methods.

As discussed in Section 3.1, states with low probability (smaller than 10^{-5}) will not be considered as initial states for constructing the next CTMC. Since all reactions in the system are considered as irreversible and the incubation time is sufficiently long, a small portion of the state space often takes a large probability. In most cases, with a 10^{-5} cut-off probability we are able to preserve at least 98% of the total probability with no more than 20 states after 6-8 iterations. Level-based RM and CTMC are applied to further reduce the state space size. To achieve a good precision, a maximum level of 100 ($N = 100$) is used in this work. Thus, given a maximum concentration M of $300nM$ (or equivalently $\approx 27,100$ molecules in a volume of 1.5×10^{-13} L), each level represents $3nM$ (271 molecules) of a DNA species.

A coarse-grained RM framework is developed aiming to capture essential reactions in each operation step. Since in this work we are interested in the formation of the target stack, each model consists of two types of reactions: a main reaction which can form the target DNA stack species, and a set of side reactions that can directly or indirectly react with the species in the main reaction, thereby reducing the production of the target stack. Let us consider the following operation sequence as an example:

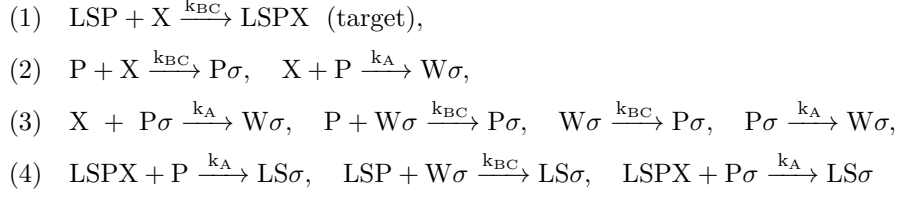
add *push* DNA data operator P ; add *dataX* DNA data operators X ; add *push* DNA data operator P

where we assume the initial species in the chemistry are bead-bound *linker-starter* LS and *starter* S. We can write a unique RM for each incubation stage after adding each new DNA data operator:

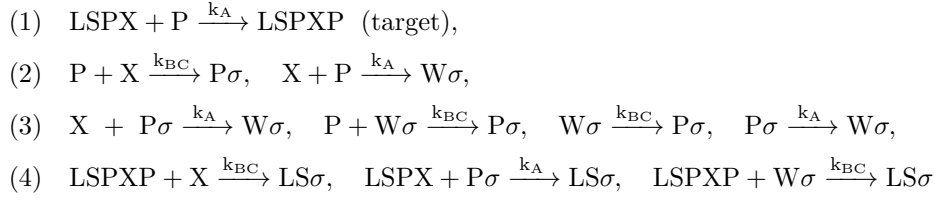
RM1 (add *push* DNA data operators P):



RM2 (add *dataX* DNA data operators X):



and RM3 (add *push* DNA data operators P):

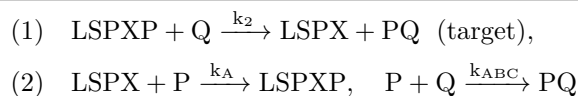


where σ denotes all possible polymers constructed by P and X (PXP, PXPX, PXPXP,...). The first reaction in each RM is the main reaction which produces target stack species (i.e., LSP in RM1, LSPX in RM2, and LSPXP in RM3), while the rest are side polymerisation reactions. In each main reaction, the initial concentrations of new-added reactant (X or P) is always higher than those pre-existing reactant (LS, LSP or LSPX) due to the wash in previous step. Therefore, a considerable amount of the newly added reactants that cannot take part in the reactions and survive from wash events would remain in the chemistry. During the incubation stage of the next operation step, these residues or the σ polymer they formed negatively affect the production rate of target stacks (i.e., side reactions (2)-(4) in RM2 and RM3). In particular, reactions (2) and (3) are hybridisation of non bead-bound supernatants to σ polymers, and reactions (4) are hybridisation of bead-bound stack and supernatants. In the model, we do not explicitly represent the detailed structure of σ polymers as it generates RM with infinite reactions. Instead, we only show their left-most (start) region which determines the capability to hybridise with the stack species in the main reactions. In this way, the RM complexity is greatly reduced. There are two types of σ polymers that can be formed under such representation: the one starting with *push* P DNA data operator ($P\sigma$) and the one starting with *data* W DNA data operator ($W\sigma$). Both of them are able to hybridise with the stack species in the main reaction inhibiting the production of target stacks.

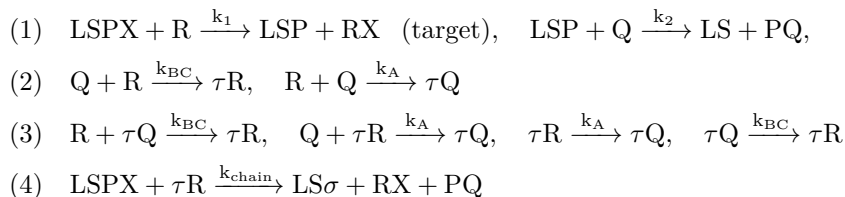
On the other hand, some reactions are not considered in our RM framework either because the molecular level of their reactants stays at zero, or they do not harm the production of target stacks. For example, two σ polymers with complementary start and end region may hybridise and form a longer polymer: $P\sigma W + P\sigma P \longrightarrow P\sigma P$. Such reaction in fact reduces the polymer concentration and thereby benefits the production of target stacks. By ignoring those reactions, we assume that our RM is always the “worst” case for the target stacks’ formation, and thus the lower bound of their concentration is guaranteed.

We can use the same idea to build RMs for the *read* and *pop* operation steps. Popping the last added *push* DNA data operator and then reading the *data*X DNA data operator yield the following RMs:

RM4 (add *pop* DNA data operators Q):



and RM5 (add *read* DNA data operators R):



Similar to the RMs for *push* and *write* steps, the production of target stack species (LSPX in RM4, and LSP in RM5) are affected by both main reactants and side reactions. The residual species surviving wash events (i.e., P in RM4, and Q in RM5) can either directly hybridise with target stack or indirectly react with them via τ polymers which trigger chain annihilation reactions [20], namely (4) in RM5. It is worth noting that the occurrence of chain annihilation reactions is determined by the right-most (end) region of τ polymer. Thus the right-most region of τ is considered explicitly during the polymer formation, which produces either τR or τQ (i.e., reactions (2) in RM5).

3.3 Model example

To illustrate how the CTMC-based DNA stack model works, consider the below operation sequence:

add *push* DNA data operator P ; add *dataX* DNA data operator X ; add *push* DNA data operator P .

The corresponding RMs are the RM1-RM3 illustrated above. To start, the initial state of RM1 is set as:

$$(\text{LS}=81, \text{P}=100, \text{S}=2, \text{LSP}=0, \text{SP}=0)$$

which corresponds to $\text{LS} = 243nM$, $\text{P} = 300nM$ and $\text{S} = 6nM$ (lower LS concentration is caused by the wash events in set-up process). The supernatant survival rate ϕ is set as 0.2, and the loss rate of bead-bound species μ is 0.1. The incubation time of each step is 30 minutes. Moreover, we only consider transient states with probability greater than 10^{-5} as initial states for the next RM. With the above conditions, a level-based CTMC is built from RM1 with 246 states and 408 transitions, and after incubation the transient probability ($> 10^{-5}$) for the following state is

$$(\text{LS}=0, \text{P}=17, \text{S}=0, \text{LSP}=81, \text{SP}=2) = 0.9999999436478932.$$

The state shows that after incubation all copies of LS contributed to forming LSP, but there are still 17 levels of P remained in the chemistry. Then the wash event is performed by removing 80% ($\phi = 0.2$) of the supernatant species P and SP, and 10% ($\mu = 0.1$) of the (bead-bound) target stack LSP.

5:10 Probabilistic Model Checking of a DNA Stack Nano-Device

To construct the next CTMC, the states are also extended with new variables representing the new species introduced by RM2. Moreover, variable X is set to 100, meaning that $300nM$ of new *dataX* DNA data operators is added to the chemistry. State variables corresponding to the species LS, S and SP will not be considered as their concentration levels will remain zero, giving the following initial state and probability:

$$(P=3, LSP=73, X=100, LSPX=0, LS\sigma = 0, P\sigma = 0, W\sigma = 0) = 0.9999999436478932.$$

The CTMC built from RM2 and the above initial state consists of 5,782 states and 20,108 transitions. The transient probability shows that after incubation there are 25 states with probability greater than 10^{-5} , with the total probability greater than 99.99%. By performing a wash event and state extension, 7 initial states are generated for building the third CTMC that adds *push* DNA data operators (RM3):

$$(P=100, LSPX=66, X=3, LSPXP=0, LS\sigma = 0, P\sigma = 0, W\sigma = 0) = 0.01657086916873019$$

$$(P=100, LSPX=65, X=3, LSPXP=0, LS\sigma = 0, P\sigma = 0, W\sigma = 0) = 0.12083719755305258$$

$$(P=100, LSPX=64, X=3, LSPXP=0, LS\sigma = 0, P\sigma = 0, W\sigma = 0) = 0.17053834871599782$$

$$(P=100, LSPX=63, X=3, LSPXP=0, LS\sigma = 0, P\sigma = 0, W\sigma = 0) = 0.03350530289916429$$

$$(P=100, LSPX=63, X=4, LSPXP=0, LS\sigma = 0, P\sigma = 0, W\sigma = 0) = 0.36343236468968576$$

$$(P=100, LSPX=64, X=4, LSPXP=0, LS\sigma = 0, P\sigma = 0, W\sigma = 0) = 0.26484767882197474$$

$$(P=100, LSPX=65, X=4, LSPXP=0, LS\sigma = 0, P\sigma = 0, W\sigma = 0) = 0.03026448265945934$$

The resulting CTMC has 46,500 states and 180,110 transitions, and completes this simple example.

4 Results

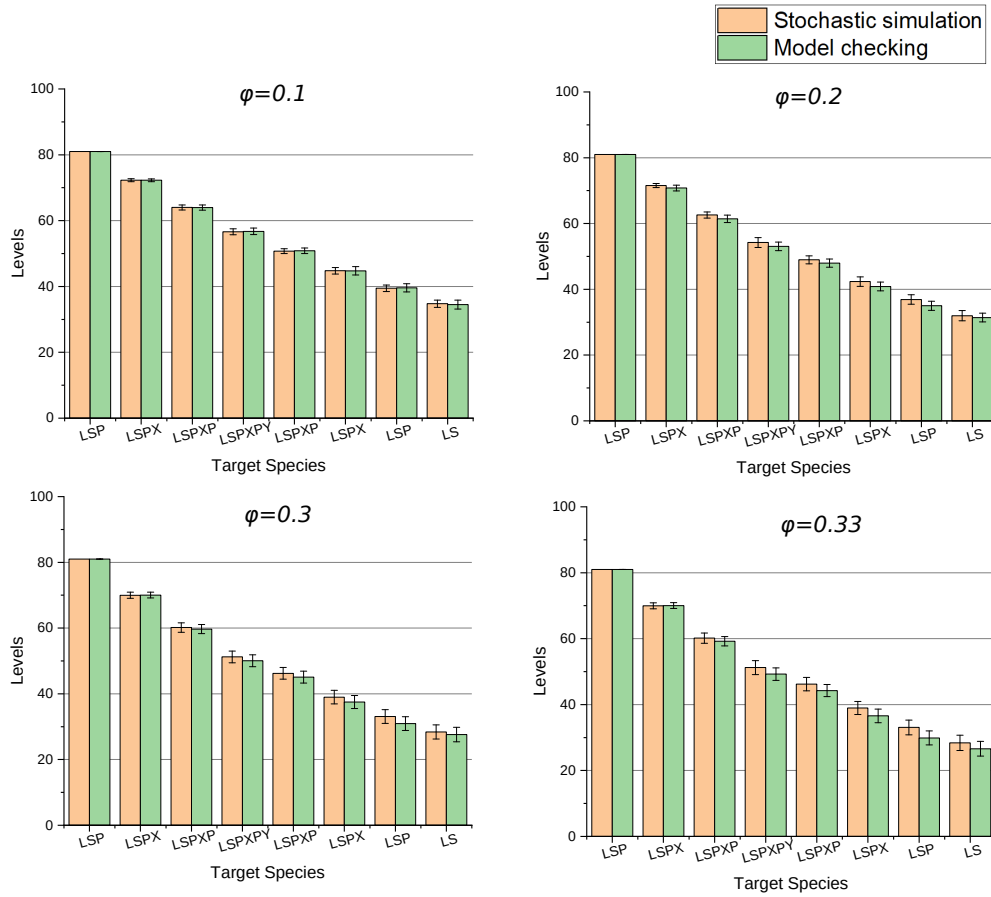
In this section, we apply our CTMC-based framework to model a 2-signal DNA stack system that manipulates two data elements, X and Y . The model will be first validated against stochastic simulation of a rule-based model implementing the same system. Then we apply probabilistic model checking technique to analyse and optimise the system.

4.1 Modelling and validating a 2-signal DNA stack system

The 2-signal DNA stack system pushes two data elements *dataX* (X) and *dataY* (Y) to the stack, and then performs two pop operations to read out the recorded data:

```
add push DNA data operators P ; add dataX DNA data operators X ; add push DNA data
operators P ; add dataY DNA data operators Y ;
add read DNA data operators R ; add pop DNA data operators Q ; add read DNA data
operators R ; add pop DNA data operators Q
```

The reaction models of the above eight operation steps are detailed in the Appendix. A maximum molecular level $N = 100$ is applied, and a fixed $300nM$ (100 levels) of each DNA data operators is added to the chemistry at the beginning of each operation step. For each step, we evaluate the mean level of the target stack after 30 minutes of incubation time.



■ **Figure 3** Comparison of the PRISM CTMC-based model and stochastic simulation of the rule-based DNA stack model: mean concentration of the target stacks after 30 minutes of incubation time for each reaction step for different values of ϕ ; $\mu = 0.1$ in all cases.

■ **Table 1** Performance of CTMC-based model and rule-based stochastic simulation: state space and transition size for all eight operation steps, total state space probability in the last step, and CPU times for model checking (MC) and stochastic simulation (SS).

	States	Transitions	Total probability	CPU time (MC)	CPU time (SS)
$\phi = 0.1$	3.3×10^4	1.1×10^5	99.87%	1.5 mins	1.8 mins
$\phi = 0.2$	2.8×10^5	1.2×10^6	99.59%	39 mins	2.2 mins
$\phi = 0.3$	2.8×10^6	1.5×10^7	98.77%	5.1 hours	2.7 mins
$\phi = 0.33$	4.0×10^6	2.2×10^7	98.43%	8.0 hours	2.9 mins

The green bars in Figure 3 show the results of the CTMC-based model with fixed $\mu = 0.1$ and various ϕ values. It can be observed that the mean level of the target stacks decrease gradually w.r.t both the operation step and the increase in ϕ . This is not only because 10% of the stack species are removed during each wash event, but also because residues (e.g., strands P, X, R) that survived the wash accelerate side reactions. As a result, the level of the target stack (LS) drops to 26 in the last step when $\phi = 0.33$ is applied. In contrast, since less residue survives when $\phi = 0.1$, there are 34 levels of the target stack (LS) produced in

the last step. Some performance statistics are reported in Table 1. The size of the state space grows significantly with ϕ , and the computation takes 8 hours when $\phi = 0.33$. Despite the large state space, in all cases more than 98% of the total probability reaches the last step when a 10^{-5} cut-off probability is used in each step.

Due to the simplifications applied to the RMs, behaviours such as polymer-polymer hybridisation are not considered in the CTMC-based model. It is therefore interesting to compare the results of our model with the results of the more detailed model. We compare our CTMC-based model against stochastic simulation of a rule-based model implementing the same system [20]. The rule-based model covers 20 DNA strand polymerisation and displacement rules. The comparison is shown in Figure 3. The stochastic simulations are run with 100 replicates for each case. Overall there is a good agreement between the two models. In the CTMC-based model, since we simplify the RMs by guaranteeing the lower bound of target stack, the mean level is always lower than that of the stochastic simulation. With $\phi = 0.1$, however, such difference is negligible (less than 1% in the last step). This is due to the small amount of residues survived from wash, which reduces the occurrence of the side reactions. With $\phi = 0.33$ more residues are in the chemistry, thus the difference between the two models increases to 4%. The performance of the stochastic simulations is given in Table 1. Although a single simulation takes 1-3 mins to finish, a much longer time (normally a few hours) is needed to collect enough replicates for computing statistically reliable results.

4.2 Optimising incubation time

Due to reaction irreversibility, the level of the target stack can eventually reach a stable value if there is no other species with which it can react. Knowing the time to reach such a stable state is important as it helps to determine the minimum incubation time. This is particularly useful for large DNA stack systems, which may involve tens of operation steps. In this section, we will analyse our probability CTMC-based model to optimise the incubation time of each DNA operation step. The 2-signal system developed in the previous section is considered. The operations of push, write, pop, and read are achieved by adding a fix amount ($300nM$) of each of the respective DNA data operators. Based on *in vitro* experiment measurements [20], ϕ and μ are set as 0.33 and 0.1, respectively.

To optimise the incubation time, we use the probabilistic model checker PRISM to quantitatively verify the following CSL property:

$$Prop1 : P_{=?}[G_{\geq T} \text{ 'target_spec_stable' }] .$$

Intuitively, the property asks: “What is the probability that the target stack reaches a stable level from time T onwards?” The condition “target_spec_stable” depends on the reactions of a given operation step. For example, for the RM2 of Section 3.2, the property becomes:

$$Prop1(RM2) : P_{=?}[G_{\geq T} (LSP = 0 \wedge P = 0 \wedge P\sigma = 0)]$$

That is, the target stack LSPX reaches its stable level when all the main reactant LSP is transformed, and there is neither *push* DNA data operators P nor $P\sigma$ polymers in the chemistry (LSP, X, P, and $P\sigma$ are the species that can affect the production of LSPX). The main reactant X is not considered in the property as it has higher level than LSP. Thus, the property can always be satisfied at some time-unit as LSP, P, and $P\sigma$ can eventually be consumed by X-ended species. Using the same idea, we can re-write *Prop1* for any reaction network. For example, the yield of target species LS in RM5 in Section 3.2 is determined by LSPX, Q and τQ . The corresponding property is:

$$Prop1(RM5) : P_{=?}[G_{\geq T} (LSPX = 0 \wedge Q = 0 \wedge \tau Q = 0)] .$$

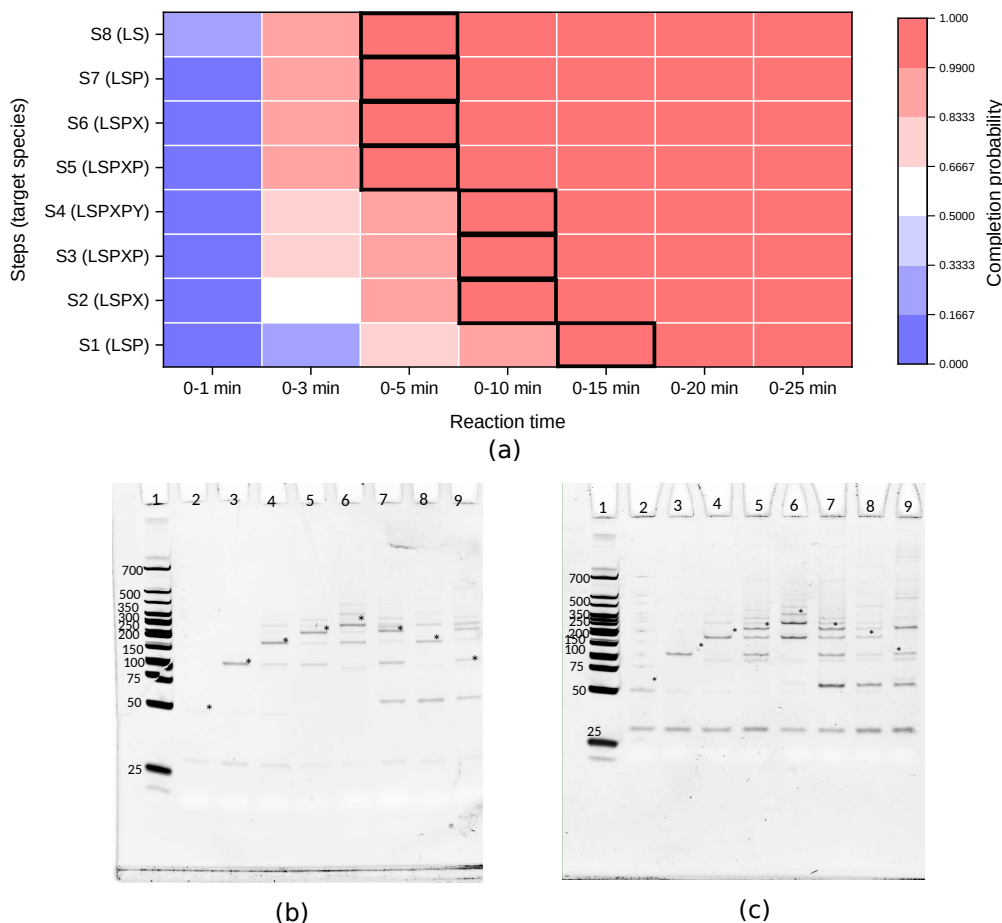


Figure 4 (a) Heatmap for the satisfaction probability of *Prop1* for target species of the 2-signal DNA stack over a range of time units T , from 60 (1 min) to 1,500 (25 min). Cells with black border indicate the suggested incubation time for each operation step. (b) Gel electrophoresis for a 2-signal DNA stack operation with a 30 minutes incubation. For each lane the target species has an asterisk next to it. Every other unlabeled species is a byproduct. Specifically: Lane 1: NEB low Molecular ladder (see Appendix for details); Lane 2: S; Lane 3: SP; Lane 4: SPX; lane 5: SPXP; Lane 6: SPXPX; Lane 7: SPXPX+R; Lane 8: SPXPX+RQ; and Lane 9: SPXPX+RQR. (c) Gel electrophoresis for a 2-signal DNA stack with the variable incubation timings taken from the schema optimised by probabilistic model checking.

Figure 4 (a) shows the probability distribution of *Prop1* with incubation time T ranging from 60 units (1 min) to 1500 units (25 min). The highlighted cells correspond to properties with the lowest reaction time T as well as having at least 99% satisfaction probability. From the result, it is suggested that an incubation time of 10-15 min is required for the first four operation steps. This is because at these stages the concentration levels of the reactants in the main reactions are relatively high and also close to each other. However, the incubation time can be reduced to 5 min in the following steps, as there is a significant difference in the levels of the reactants, which causes a rapid consumption of reactants with lower concentration.

Experimental validation of the model checking result is given in Figure 4 (b) and (c). A full stack operation for the 2-signal system is performed with 30 minutes incubation time (Figure 4 (a)), and the variable incubation time suggested from the model checking

((Figure 4 (b)). As can be seen, in addition to the target species at each operation stage (asterisks), multiple unwanted species also exist. This is in particular the case during the reading process (lanes 7-10) with the increase of side reaction rates. However, the two images show a qualitative agreement, indicating that a similar performance can be achieved with optimised incubation time.

4.3 Improving target stack yield

The results in Section 4.1 show that high ϕ values (fraction of supernatants surviving the washing) can significantly reduce the yield of the target stack. This is due to the large amount of undesired DNA species (e.g., *push* DNA data operators P in RM2) that survives from the wash event and increase the rate of side reactions, which compete with the main reactions. Enhancing wash efficiency and decreasing ϕ is difficult to achieve *in vitro*. However, we may still be able to minimise the concentration of undesired species by controlling the amount of each new DNA data operator added to the chemistry at each step. Consider the 2-signal system with $\phi = 0.33$, where a fixed $300nM$ of each DNA data operators is added at each step. The main reaction in the second step “*adding dataX*” is $LSP + X \xrightarrow{K_{BC}} LSPX$. Initially, there is around $210nM$ of LSP in the system. Adding $300nM$ of *dataX* DNA data operators results in a large portion of these new DNA data operators not being able to react with LSP. Instead, they become undesired species in the next step, which in turn negatively affects the yield of the target stack LSPXP. Therefore, if we reduce the amount of these new *dataX* DNA data operators, a higher yield of the target stack would be expected.

In this section, we employ the CSL logic and PRISM to maximise the yield of the target stack by tuning the amount of each DNA data operator added at each step. The idea is to first compute a maximum target stack yield when adding a sufficient amount of the DNA data operators (in our case, $300nM$). Taking the result as a reference value, we gradually decrease the amount of DNA data operators added until reaching a minimum value for which the system can still produce the desired level of target stack. We use the following CSL property:

$$Prop1(C) : P_{\geq 0.99} [F[1800, 1800] \text{ target_spec} \geq C] .$$

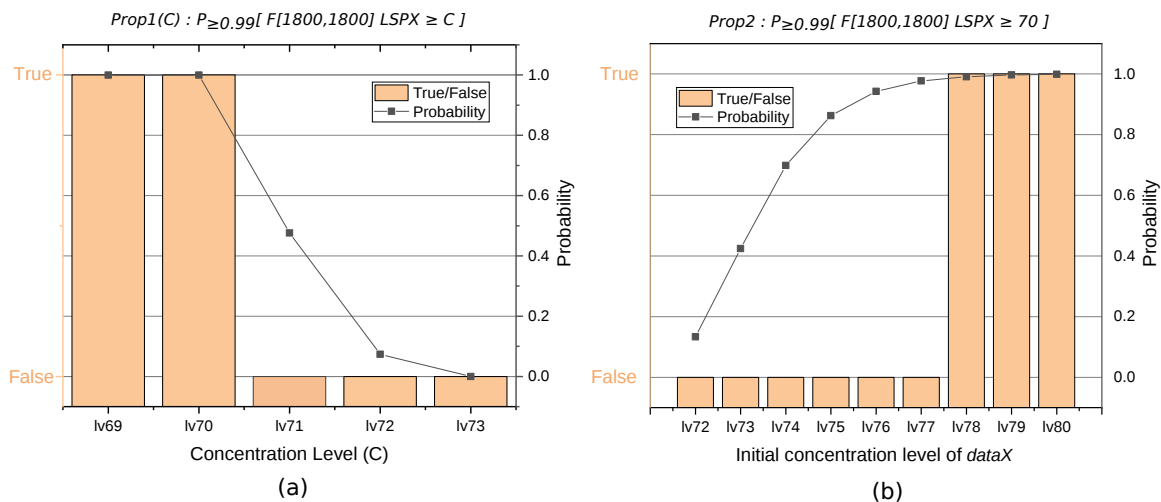
Informally, $Prop1(C)$ means that “Given a molecular level C , the probability of target stack reaching such level after 30 minutes is greater than 99%.” We apply the property for $1 \leq C \leq 100$, and take the maximal C which satisfies the property as the reference level C_{ref} :

$$C_{ref} = \text{Max}\{C \mid Prop1(C) \wedge 1 \leq C \leq 100\} .$$

Figure 5(a) gives an example of determining the reference level for the step “*adding dataX*” (RM3). When adding 100 levels ($300nM$) of *dataX*, the property $Prop1(C) : P_{\geq 0.99} [F[1800, 1800] LSPX \geq C]$ is *false* for $C = 71$ ($213nM$) and beyond, and is *true* for $C = 70$ ($210nM$) and below, indicating that the system can produce a maximum level of 70 LSPX with 99% probability. Thus we take 70 as the reference level C_{ref} . Now, the following CSL property is employed to find the optimised addition concentration:

$$Prop2 : P_{\geq 0.99} [F[1800, 1800] \text{ target_spec} \geq C_{ref}] .$$

The property evaluates whether the system can yield with high probability ($\geq 99\%$) at least C_{ref} levels of the target stack at 30 minutes. In order to find the optimal level, we model check the system with different initial states (i.e., smaller levels) until $Prop2$ is false. In this



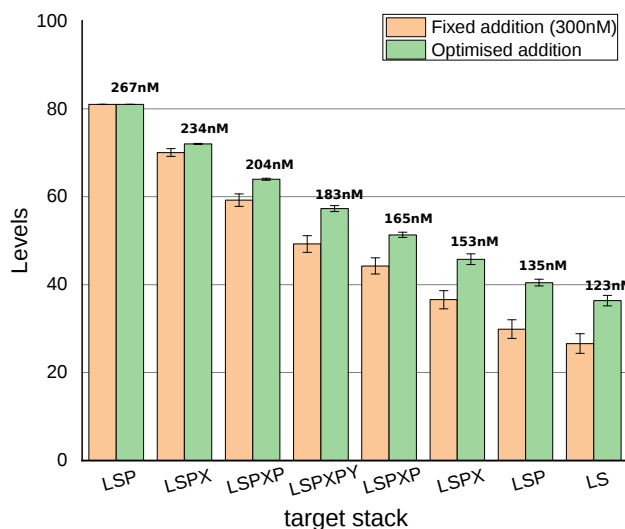
■ **Figure 5** Optimising DNA addition levels (step “adding dataX” (RM3)) using probabilistic model checking. Panels: (a) Evaluating $Prop1(C)$ for $C = \{60, 70, 71, 72, 73\}$ when adding $300nM$ dataX; (b) Evaluating $Prop2$ with reduced initial level of dataX, from 72 to 80. Yellow bars: satisfaction of properties; lines: satisfaction probabilities.

way we find the minimal initial level that satisfies the reference level of target stack C_{ref} . In Figure 5(b) we evaluate $Prop2$ using the reference level obtained in (a) for a range of initial levels. The result suggests that adding 78 levels ($234nM$) of dataX is the minimum value for which the system produces the same amount of LSPX when adding $300nM$ of dataX. The optimised value, however, can avoid up to $64nM$ of dataX remaining in the chemistry, which can harm the system yield in the next step.

We applied our approach to optimise the 2-signal system with $\phi = 0.33$ and $\mu = 0.1$. We compare the mean level of target stacks in the optimised system with the standard system with fixed concentrations, as shown in Figure 6. In the last step, the yield of target stack in the optimised system is 40% higher than the yield of the standard system.

5 Conclusion and future work

In this work, we applied probabilistic model checking to model and optimise a DNA stack nano-device. An iterative CTMC-based workflow is developed within the PRISM probabilistic model checker to model the system. Each iteration describes a single DNA operation step: add new DNA data operator, incubation, and wash. The initial state of the $(i + 1)^{th}$ step is determined by the model checking result (transient probability) of the i^{th} step. Level-based CTMCs and simplified reaction models are applied to reduce the state space size. We have used our framework to develop a 2-signal DNA stack system model. The model is validated against stochastic simulation of a rule-based model implementing the same system. We use probabilistic model checking to formally analyse and optimise our 2-signal system. Results suggest that when a fix amount ($300nM$) of each DNA data operator are added in each operation step, an incubation time of 10-15 minutes is required for the first several operation steps, while for the following steps the incubation time can be reduced to 5 minutes. We have experimentally validated the result by comparing the gel electrophoresis obtained from the optimised and unoptimised (30 min incubation time) systems. We also applied probabilistic



■ **Figure 6** Mean level of target stacks after adding a fixed 300nM of each DNA data operator in each operation step (yellow bars) vs. adding the optimised level (green bars). The values on top of the green bars are the actual (optimised) levels added to the systems.

model checking to maximise the yield of target stacks by optimising the amount of DNA data operators added at each reaction step: our optimised model shows a 40% increase in target stack yield.

Experimental validation on improving the yield of DNA nanostructures is an ongoing work. The major challenges of the validation are two-fold. First, the concentration difference between the optimised and unoptimised protocols in some cases (i.e., LSP and LSPX in Figure 6) is small, which makes the analysis of the results difficult. For example, it is hard to figure out whether the differences in the experimental output are caused by the model creating the correct schematic or the stochastic nature of pipetting error. Second, the model is influenced by two main parameters: ϕ and μ . Both these factors exhibit a large influence over the model and are difficult to optimise and control in the experimental setting. There are two strategies that we could implement in future to potentially solve these problems. One approach to explore the pipetting stochasticity would be to use a liquid handling robot to carry out the operations instead of a manual pipette, as robots have much higher accuracy than human operators and are not prone to fatigue and other factors. It may also improve the bead loss and washing efficiency. Another strategy to mitigate the bead loss could be to use another DNA nanostructure to tether the device instead of beads. Finally, an enzymatic wash could be explored to reduce the noise and in turn the off target species by digesting potential byproducts and leaving on the target species intact.

References

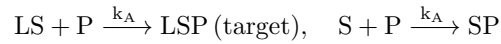
- 1 Leonard M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024, 1994. doi:10.1126/science.7973651.
- 2 Oana Andrei and Muffy Calder. A Model and Analysis of the AKAP Scaffold. *Electronic Notes in Theoretical Computer Science*, 268:3–15, 2010. Proceedings of the 1st International Workshop on Interactions between Computer Science and Biology (CS2Bio’10). doi:10.1016/j.entcs.2010.12.002.
- 3 Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert K. Brayton. Model-checking continuous-time markov chains. *ACM Trans. Comput. Log.*, 1(1):162–170, 2000. doi:10.1145/343369.343402.

- 4 Christel Baier, Edmund M. Clarke, Vasiliki Hartonas-Garmhausen, Marta Z. Kwiatkowska, and Mark Ryan. Symbolic model checking for probabilistic processes. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 430–440. Springer, 1997. doi:10.1007/3-540-63165-8_199.
- 5 Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003. doi:10.1109/TSE.2003.1205180.
- 6 Muffy Calder, Vladislav Vyshemirsky, David Gilbert, and Richard Orton. Analysis of Signalling Pathways Using Continuous Time Markov Chains. In Corrado Priami and Gordon Plotkin, editors, *Transactions on Computational Systems Biology VI*, pages 44–67. Springer Berlin Heidelberg, 2006. doi:10.1007/11880646_3.
- 7 Luca Cardelli. Strand algebras for DNA computing. In Russell J. Deaton and Akira Suyama, editors, *DNA Computing and Molecular Programming, 15th International Conference, DNA 15, Fayetteville, AR, USA, June 8-11, 2009, Revised Selected Papers*, volume 5877 of *Lecture Notes in Computer Science*, pages 12–24. Springer, 2009. doi:10.1007/978-3-642-10604-0_2.
- 8 Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8, September 2013. doi:10.1038/nnano.2013.189.
- 9 George M. Church, Yuan Gao, and Sriram Kosuri. Next-Generation Digital Information Storage in DNA. *Science*, 337(6102):1628–1628, 2012. doi:10.1126/science.1226355.
- 10 Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 2001.
- 11 Parker Dave, Norman Gethin, and Kwiatkowska Marta. Prism user manual, 2017. URL: <http://www.prismmodelchecker.org/manual/>.
- 12 Harold Fellermann, Annunziata Lopiccio, Jerzy Kozyra, and Natalio Krasnogor. In vitro implementation of a stack data structure based on DNA strand displacement. In Martyn Amos and Anne Condon, editors, *Unconventional Computation and Natural Computation - 15th International Conference, UCNC 2016*, volume 9726 of *Lecture Notes in Computer Science*, pages 87–98. Springer, 2016. doi:10.1007/978-3-319-41312-9_8.
- 13 Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976. doi:10.1016/0021-9991(76)90041-3.
- 14 John Heath, Marta Kwiatkowska, Gethin Norman, David Parker, and Oksana Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 391(3):239–257, 2008. doi:10.1016/j.tcs.2007.11.013.
- 15 Savas Konur, Marian Gheorghe, Ciprian Dragomir, Laurentiu Mierla, Florentin Ipate, and Natalio Krasnogor. Qualitative and quantitative analysis of systems and synthetic biology constructs using P systems. *ACS Synthetic Biology*, 4(1):83–92, 2015. doi:10.1021/sb500134w.
- 16 Jerzy Kozyra, Harold Fellermann, Ben Shirt-Ediss, Annunziata Lopiccio, and Natalio Krasnogor. Optimizing nucleic acid sequences for a molecular data recorder. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17*, pages 1145–1152. ACM, 2017. doi:10.1145/3071178.3071345.
- 17 Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In Marco Bernardo and Jane Hillston, editors, *Formal Methods for Performance Evaluation: 7th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2007, Bertinoro, Italy, May 28-June 2, 2007, Advanced Lectures*, pages 220–270. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-72522-0_6.
- 18 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591. Springer, 2011. doi:10.1007/978-3-642-22110-1_47.

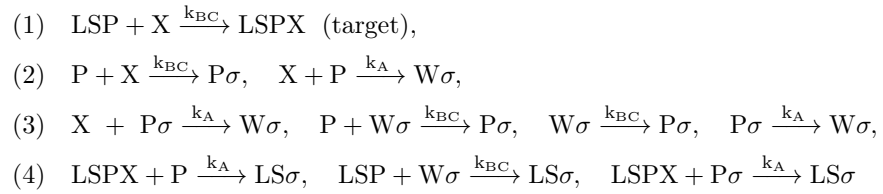
- 19 Matthew R. Lakin, David Parker, Luca Cardelli, Marta Kwiatkowska, and Andrew Phillips. Design and analysis of dna strand displacement devices using probabilistic model checking. *Journal of The Royal Society Interface*, 9(72):1470–1485, 2012. doi:10.1098/rsif.2011.0800.
- 20 Annunziata Lopiccolo, Ben Shirt-Ediss, Emanuela Torelli, Abimbola Feyisara Adedeji Olulana, Matteo Castronovo, Harold Fellermann, and Natalio Krasnogor. A last-in first-out stack data structure implemented in DNA. *Nature Communications*, 12(1):4861, 2021. doi:10.1038/s41467-021-25023-6.
- 21 Lulu Qian, David Soloveichik, and Erik Winfree. Efficient Turing-Universal Computation with DNA Polymers. In Yasubumi Sakakibara and Yongli Mi, editors, *DNA Computing and Molecular Programming*, pages 123–140. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-18305-8_12.
- 22 Lulu Qian and Erik Winfree. A Simple DNA Gate Motif for Synthesizing Large-Scale Circuits. In Ashish Goel, Friedrich C. Simmel, and Petr Sosik, editors, *DNA Computing and Molecular Programming*, pages 70–89. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-03076-5_7.

A Complete reaction models for 2-signal DNA stack system

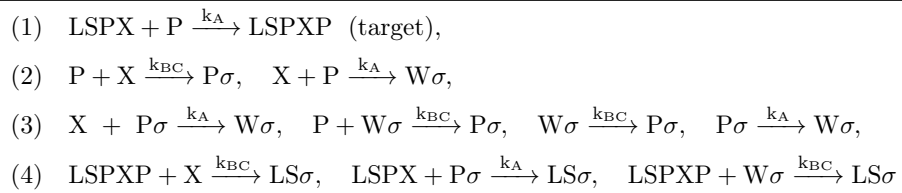
RM1 (add *push* data operators P):



RM2 (add *dataX* data operators X):

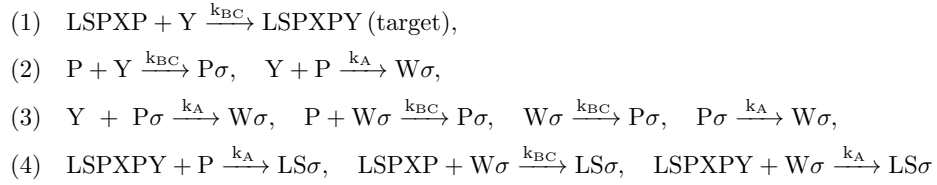


RM3 (add *push* data operators P):

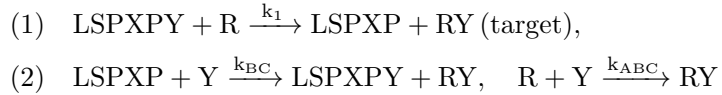


where the bi-molecular rate constants were found to be approximately $K_A \approx K_{BC} \approx 3 \times 10^4 \text{Ms}^{-1}$, and $K_{ABC} \approx 2.5 \times 10^5 \text{M}^{-1}\text{s}^{-1}$. The bi-molecular strand displacement rate constants K_1 and K_2 were assumed equal to hybridisation constants K_A and K_{BC} respectively, due to the long 28nt “toeholds” of the strand displacement reaction [20]. Moreover, the chain annihilation reactions were approximated as a single step bimolecular reaction whose rate constant was set to the rate constant of the initial hybridisation event as an upper bound i.e. $k_A^{chain} = k_A$ and $k_{BC}^{chain} = k_{BC}$.

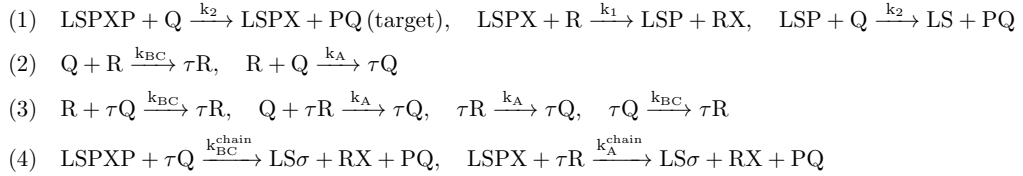
RM4 (add *dataY* data operators Y):



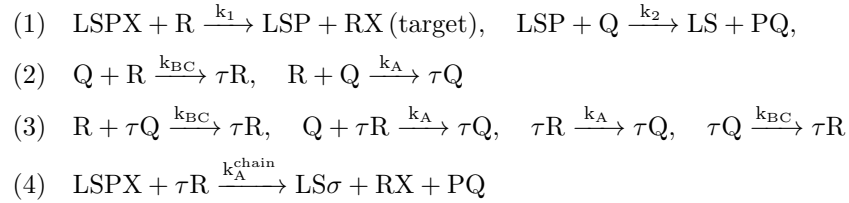
RM5 (add *read* data operators R):



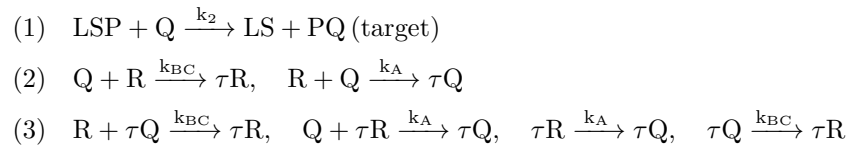
RM6 (add *pop* data operators Q):



RM7 (add *read* data operators R):



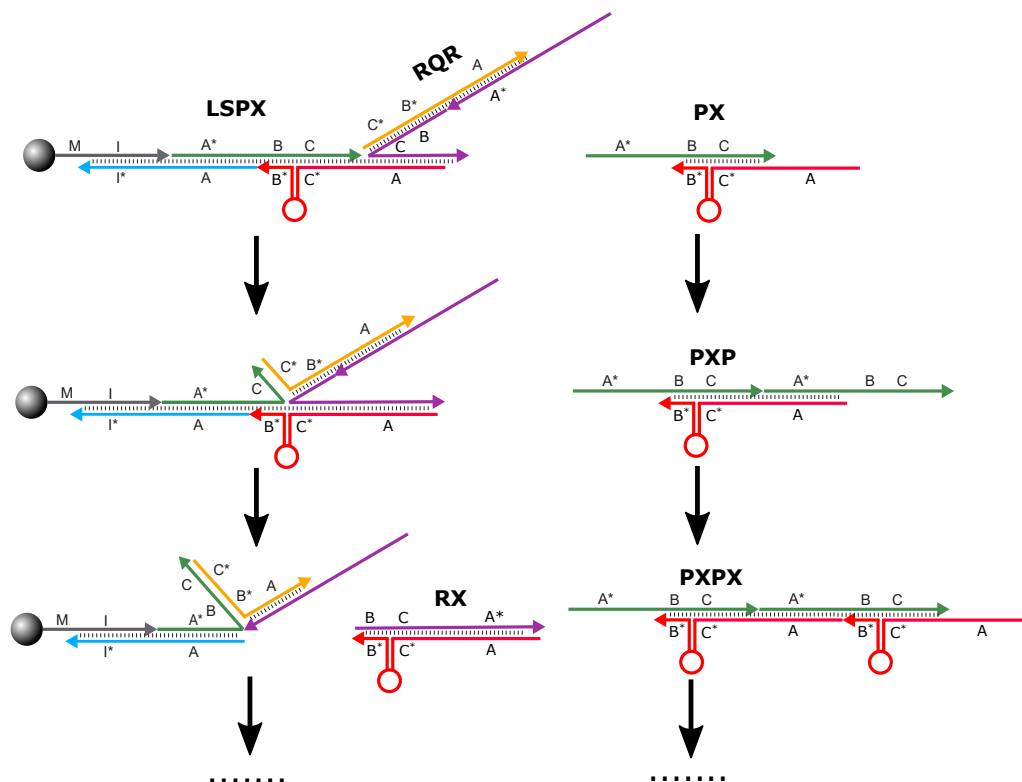
RM8 (add *pop* data operators Q):



B Determine initial state of CTMC for RM1

The initial state for generating CTMC from the first reaction model is assumed the following steps have been performed:

- 100 levels of (300nM) *linker* DNA data operators incubate with 100ul sepharose beads for sufficient time. It's assumed that all the DNA data operators can irreversible bind to beads which yields 100 levels of bead-bounded L stacks.
- A wash event is perform by assuming μ percent of the L stacks is removed. Given $\mu = 0.1$, 90 levels of L remain in the chemistry after the wash.



■ **Figure 7** Examples of a hypothesised “chain annihilation” multi-stage reaction (left), and P, X polymer formation (right).

- 100 levels of *start* DNA data operators are added to hybridise with L. It is assumed that the irreversible linker-start reaction proceeded to 100% completion over the incubation time producing 90 levels of LS with 10 levels of supernatant S remain in the chemistry.
- A second wash event is performed by removing 10% LSP stacks, and 80% S (assuming $\phi = 0.2$). The result (i.e., LS = 81, S = 2) will be used as the initial state in the first CTMC.

C Experimental Validation

■ **Table 2** DNA sequences for in vitro validation [20].

Strand	Symbol	Length	Domains	Primary Sequence 5' - 3'
Start	S	50	A I*	CACACTATTTCCCTTCTACCCGCCCTATCTCATCTCTCATCTCTTAA
Push	P	56	A* BC	ATAGGGCGGGTAGAAGGGAAATAGTGTGATCCAGTTATT ATAGTTTTGAAGCGTAT
Write	x	56	A C*B*	CACACTATTTCCCTTCTACCCGCCCTATATACGCTTCAA ACTATAATAACTGGAT
Read	r	56	B CA*	ATCCAGTTATTATAGTTTTGAAGCGTATATAGGGCGGGTA GAAGGGAAATAGTGTG
Pop	q	56	C* B* A	ATACGCTTCAAACATAATAACTGGATCACACTATTTCCC TTCTACCCGCCCTAT
Linker	k	33	ml	GAGAGAGATGATTAAGATGAGATGAGAGATGAG
Releaser	z	33	I*m*	CTCATCTCTCATCTCATCTTAATCATCTCTCTC

For the sake of simplification, we only use data X to build stack in the experiments. This is because stacks are identified by their length, and it's unable to distinguish X stack (e.g, LSPXPX) and Y stack (e.g, LSPXPY) at the current settings as they are different in hairpin only. Such simplification, however, would not affect our model checking results, as the model excludes the situation that non-target stack becomes target stack in the later operation steps.

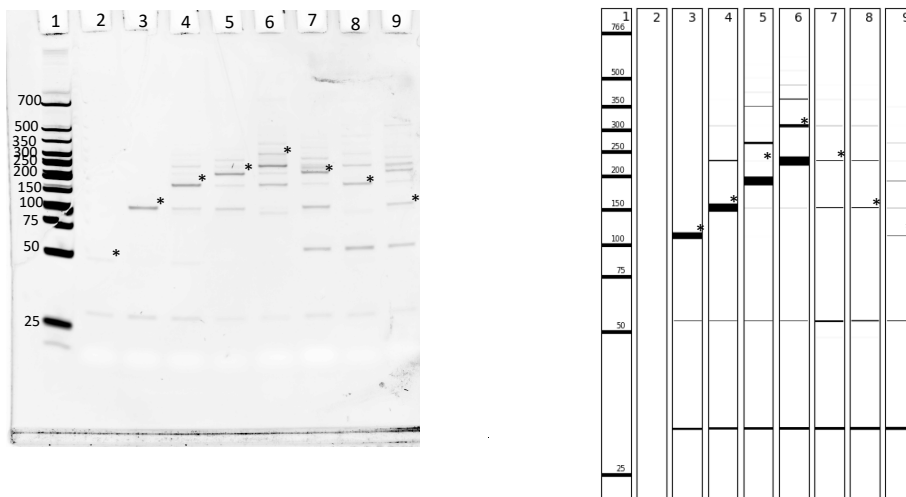
The Vgel simulator [20] is used to create a computer-generated image of what the DNA stack model output would like post releasing the stack from the beads. An example Vgel image is given in Figure 8. This Vgel can be combined with the species output to create a labelled image as we know the lengths of the strands when combined together to be: when signals are being pushed to the stack

- SP: $50 + 56 = 106$
- SPX: $50 + 56 + 56 = 162$
- SPXP: $50 + 56 + 56 + 56 = 218$
- SPXPX: $50 + 56 + 56 + 56 = 274$

When signals are being read from the stack or popped the length should drop by 56

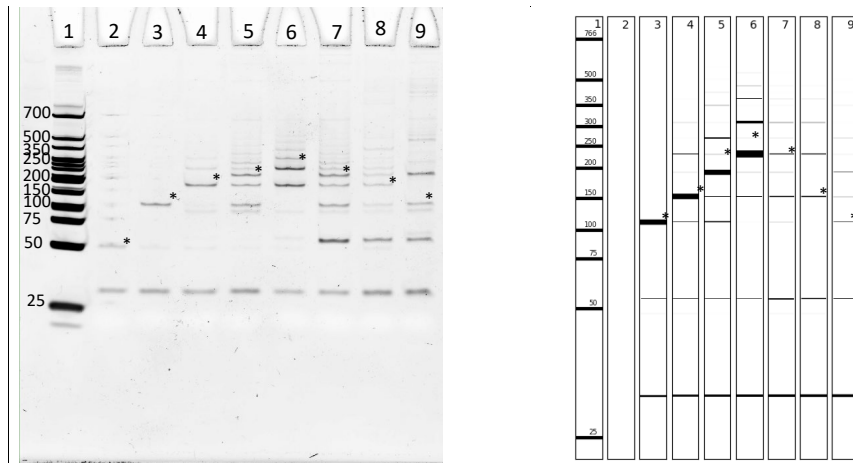
- SPXPXR: $274 - 56 = 218$
- SPXPXRQ: $218 - 56 = 162$
- SPXPXRQR: $162 - 56 = 106$

Note that Linker is not part of this calculation as this creates its own complex of length 33. There are also other complexes which are formed by side reactions.



■ **Figure 8 Full stack operation 30 minutes.** A comparison of the gel electrophoresis and Vgel for a full stack operation with a 30 minute incubation. For each lane the target species has an asterisk next to it. Every other unlabeled species is a byproduct. Gel electrophoresis image: Lane 1: NEB low molecular ladder; Lane 2: S; Lane 3: SP; Lane 4: SPX; Lane 5: SPXP; Lane 6: SPXPX; Lane 7: SPXPXR; Lane 8: SPXPXRQ; and Lane 9:SPXPXRQR. Electrophoresis conditions: Gel:10 % Novex TBE Gel 1X TBE @200V for 35minutes Staining dye: Sybr Gold Concentrations: all DNA data operators were nominally 300nM Vgel image:Lane 1: NEB low molecular ladder. Lane 2: S; Lane 3: SP; Lane 4: SPX; Lane 5: SPXP; Lane 6: SPXPX; Lane 7: SPXPXR; Lane 8: SPXPXRQ; and Lane 9: SPXPXRQR.

5:22 Probabilistic Model Checking of a DNA Stack Nano-Device



■ **Figure 9 Variable timing as per the generated schema.** A comparison of the gel electrophoresis and Vgel for a full stack operation with the incubation timings taken from the schema supplied by the model. For each lane the target species has an asterisk next to it. Every other unlabeled species is a byproduct. Gel electrophoresis image: Lane 1: NEB low molecular ladder; Lane 2: S; Lane 3: SP; Lane 4: SPX; Lane 5: SPXP; Lane 6: SPXPX; Lane 7: SPXPXR; Lane 8: SPXPXRQ; and Lane 9: SPXPXRQR. Electrophoresis conditions: 10% Novex TBE Gel in a buffer of 1X TBE @200V for 35minutes. Staining dye: Sybr Gold Concentrations:all DNA data operators were nominally 300nM Vgel image: Lane 1: NEB low molecular ladder; Lane 2: S; Lane 3: SP; Lane 4: SPX; Lane 5: SPXP; Lane 6: SPXPX; Lane: 7: SPXPXR; Lane 8: SPXPXRQ; and Lane 9: SPXPXRQ.