DNA walker circuits: computational potential, design, and verification

Frits Dannenberg[1], Marta Kwiatkowska[1],
Chris Thachuk[1], and Andrew Turberfield[2]

[1] University of Oxford, Department of Computer Science
[2] University of Oxford, Department of Physics, Clarendon Laboratory

Abstract. Unlike their traditional, silicon counterparts, DNA computers have natural interfaces with both chemical and biological systems. These can be used for a number of applications including the precise arrangement of matter at the nanoscale and the creation of smart biosensors. Like silicon circuits, DNA strand displacement systems (DSD) can evaluate non-trivial functions. However, these systems can be slow and are susceptible to errors. It has been suggested that localised hybridization reactions could overcome some of these challenges. Localised reactions occur in DNA 'walker' systems which were recently shown to be capable of navigating a programmable track tethered to an origami tile. We investigate the computational potential of these systems for evaluating Boolean functions. DNA walkers, like DSDs, are also susceptible to errors. We develop a discrete stochastic model of DNA walker 'circuits' based on experimental data, and demonstrate the merit of using probabilistic model checking techniques to analyse their reliability, performance and correctness.

1 Introduction

The development of simple biomolecular computers is attractive for engineering and health applications that require *in vitro* or *in vivo* information processing capabilities. DNA computing models which use hybridization and strand displacement reactions to perform computation have been particularly successful. DNA strand displacement systems (DSD) have been shown experimentally to simulate logic circuits [12, 11] and are known to be Turing-universal [10]. However, computing with biomolecules creates many challenges. For example, reactions within a DSD are global in the following sense: strands which are intended to react must first encounter one another in a mixed solution. The mixing of all reactants may lead to unintended reactions between strands. These systems do not, at present, ensure the spatial locality typical of other computing models. Qian and Winfree suggested that tethering DNA based circuits to an origami tile could overcome some of these challenges [11]. This idea was explored and expanded upon by Chandran *et al.* [5], who investigate how such systems could be realised experimentally, give constructions of composable circuits, and propose a biophysical model for verification of tethered, hybridization-based circuits. Our work is largely inspired by theirs, but we consider another setting which also exhibits localised reactions: DNA walker systems [2, 7, 9, 14, 13, 15].

Various DNA walkers have been experimentally realised — see [13] and references therein. Single-legged DNA walkers were recently shown capable of navigating a programmable track of strands, called *anchorages*, that are tethered to a DNA origami

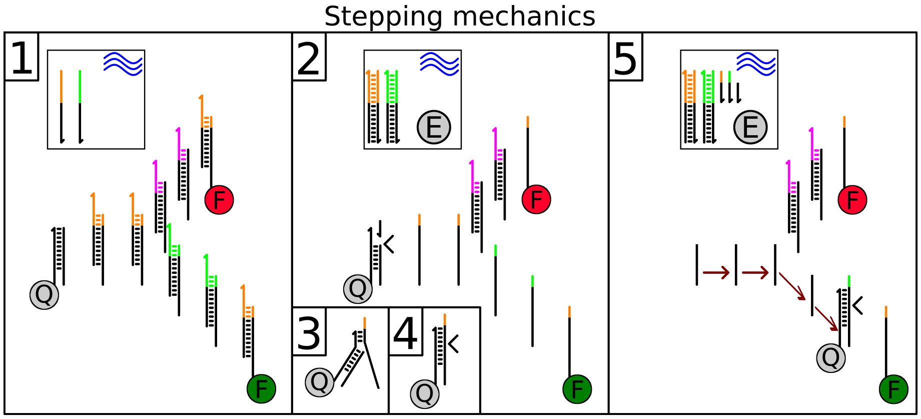


Fig. 1. (1) The walker strand carries a load (Q) that will quench fluorophores (F) when nearby. The walker is attached to the initial anchorage and all other anchorages are blocked. By adding unblocking strands, the selected track becomes unblocked. (2) The nicking enzyme (E) attaches to the walker-anchorage complex, and cuts the anchorage. The anchorage top melts away from the walker, exposing 6 nucleotides as a toehold. (3) The exposed toehold becomes attached to the next anchorage. (4) In a displacement reaction, the walker becomes completely attached to the new anchorage. The stepping is energetically favourable, because it re-forms the base pairs that were lost after the previous anchorage was cut. (5) Repeating this process, the walker arrives at a junction. The walker continues down the unblocked track, eventually reaching the final anchorage and quenching the fluorophore.

tile [13]. Movement of the walker between anchorages is shown in Fig. 1. Initially, all tracks are blocked by hybridization to blocker strands. Anchorages and their blockers are addressed by means of distinct toehold sequences (shown coloured): anchorages are selectively unblocked by adding strands complementary to their blockers as *input*. Much like field programmable gate arrays, these systems are easily reconfigured. By using programmable anchorages at track junctions, Wickham *et al.* [13] demonstrate that a walker can be directed to any leaf in a complete two-level binary tree using input strands that unblock the intended path.

In Section 2, the computational expressiveness of such walker systems is explored, using a theoretical framework that assumes ideal conditions. We highlight significant limitations of current walker systems and motivate future work. In Section 3 we develop a probabilistic model to analyse the impact of different sources of error that arise in experiments on reliability, performance and correctness of the computation. The model can be used to support the design and verification of DNA walker *circuits*.

2 Computational potential of DNA walker circuits

In this section we explore the computational potential of DNA walker systems. We focus on deterministic Boolean function evaluation and call the resulting constructions *DNA walker circuits*. We begin by defining a model of computation that makes explicit the underlying assumptions that characterize the DNA walker systems considered here. These assumptions are consistent with current published experimental systems: in particular, we do not explore the potential for multiple walkers to interact within the same circuit. However, we do consider the potential consequences for parallel computation.

A *DNA walker circuit* is composed of straight, undirected, *tracks*, and *gates* that connect at most three tracks. A gate can have at most one Boolean *guard* for each track that it connects. A track adjacent to a gate is *blocked* if it has a guard that evaluates to false and is unblocked otherwise. We define a *fork gate* as having at most one input track, and exactly two guarded output tracks. Each circuit has one *source*–a fork gate with no input track denoting the initial position of a walker. A *join gate* with an output track has at most two guarded input tracks. A join gate with no output track is a *sink* and has at most three (unguarded) input tracks. Each circuit has one or more *true sink*s and one or more *false sink*s.

In a circuit $\mathcal{C}$ with Boolean guards over n variables, a *variable assignment* A for $\mathcal{C}$ is a truth assignment of those n variables. Consider any DNA walker circuit $\mathcal{C}$ and variable assignment A for $\mathcal{C}$. Let $\mathcal{C}[A]$ denote the set of reachable paths originating from the source of $\mathcal{C}$, after all guards are evaluated as blocked or unblocked, under assignment A. We say that $\mathcal{C}$ is *deterministic* under assignment A if there is exactly one path from the source to a sink in $\mathcal{C}[A]$. Note that this definition of determinism precludes the possibility of a *deadlock*, (*i.e.,* when no path from the source can reach a sink). Let $\texttt{VALUE}\,(\mathcal{C}[A])$ be the *output value* of the circuit under assignment A (*i.e.,* whether the reachable sink is a true sink or a false sink). Circuit $\mathcal{C}$ is *deterministic* if it is deterministic under all possible variable assignments.

A *circuit set* S, consisting of one or more unconnected circuits, is deterministic if and only if $\texttt{VALUE}\,(\mathcal{C}_i[A]) = \texttt{VALUE}\,(\mathcal{C}_j[A])$, for each $\mathcal{C}_i, \mathcal{C}_j \in \mathsf{S}$, under any possible assignment A. Let $\texttt{VALUE}\,(\mathsf{S}[A])$ be the value of S under assignment A. The *size* of S, denoted by $\texttt{SIZE}\,(\mathsf{S})$, is the total count of component gates.[3] We define the worst case *time* of a computation in S, denoted by $\texttt{TIME}\,(\mathsf{S})$, as the longest reachable path from a source to a sink. This notion of time captures the ability of multiple walkers to simultaneously traverse disjoint paths (one per unconnected circuit).

Let $\mathsf{S}[A]$ denote the set of reachable paths in S under assignment A (one per unconnected circuit). Given a circuit $\mathcal{C}_i \in \mathsf{S}$, we say that a gate $G \in \mathcal{C}_i$ is *reachable* in $\mathsf{C}_i[A]$ (equivalently $\mathsf{S}[A]$) if there exists an unblocked path from the source of $\mathcal{C}_i$ to G. Note that if every gate is reachable, this implies that every output track of a gate can be traversed under some variable assignment. We call gates where this is not true *redundant*. We will reason about circuit sets where all gates are reachable and non-redundant under some variable assignment. When this is not the case, the circuit set can be simplified to one that is logically equivalent.

2.1 Reporting output in DNA walker circuits

Output of a DNA walker circuit can be reported with the use of different coloured (spectrally resolvable) fluorophores and also quenchers. If a walker carries a quencher cargo, then it has the potential to decrease one of a number of different fluorescent signals from fluorophores positioned at the circuit sinks. This scenario is illustrated in Fig. 2 (Left). In a circuit to decide a Boolean function, a single, quenching, walker can only decrease the signal of a particular colour (corresponding to a particular fluorophore) by

[3] We do not investigate circuit *area* in this paper.

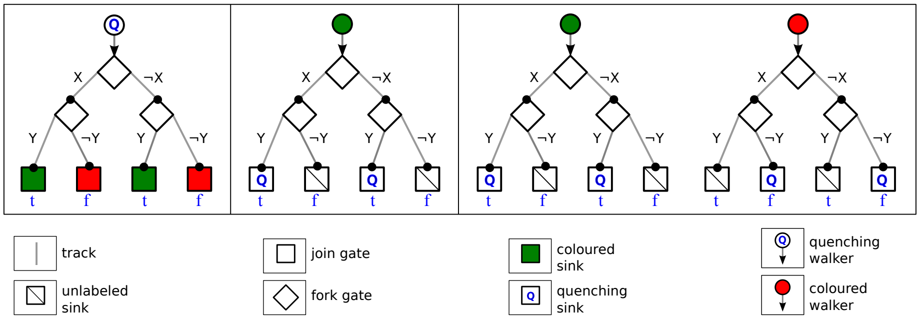


Fig. 2. Reporting Boolean decisions with DNA walker circuits. (Left) A quenching walker with red fluorophores labelling false sinks and green fluorophores labelling true sinks. A drop in signal for one colour indicates the truth value of the circuit. However, the signal drop is inversely proportional to the number of sinks of the same colour. (Center) A green coloured walker and quenching true sinks. When the circuit evaluates to true the green signal is fully suppressed. However, the fluorescence output from this circuit cannot distinguish between an incomplete computation and a false one. (Right) Two parallel copies of the circuit, with different fluorophores labelling the walkers and with quenching true sinks in one and quenching false sinks in the other: the computation is complete and unambiguously reported when one colour is suppressed.

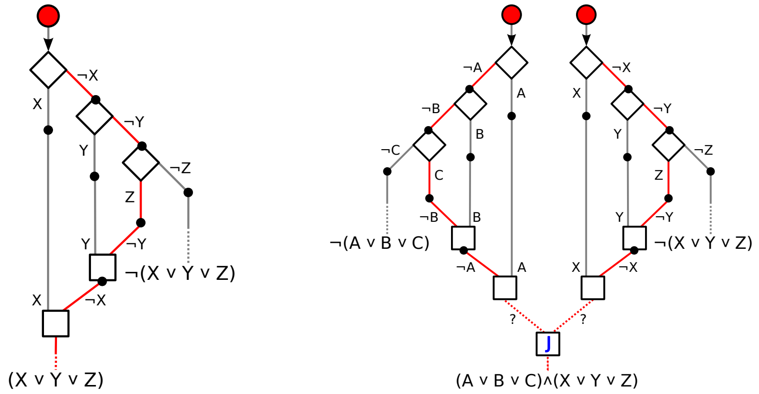
an amount that is inversely proportional to the number of sinks labelled with that same colour. Accurate output reporting could be problematic in larger circuits with many sinks. We will therefore focus only on reporting strategies that fully suppress a particular colour. Rather than carrying a quencher, a walker instead carries a fluorophore of a single colour and either all true sinks or all false sinks are labelled with quenchers. An example with quenching true sinks is shown in Fig. 2 (Center). This circuit can fully suppress the fluorophore signal when it evaluates to true, regardless of its size. However, this is a one-sided reporting strategy as one cannot distinguish between the case of an incomplete computation or one evaluating to false. As illustrated in Fig. 2 (Right) this shortcoming can be addressed by using two circuits in parallel with each using a one-sided reporting strategy. Each of the two (otherwise identical) circuits uses a different coloured walker: one has quenching false sinks and the other quenching true sinks. In this circuit set, one colour will be fully suppressed when it is true, the other when it is false, and neither will be suppressed until the computation completes.

2.2 Deterministic fork and join gates in DNA walker circuits

If all gates in a circuit set S are deterministic, it follows that S is deterministic. The following theorem shows that deterministic fork gates must have output guards that are negations of each other.

Theorem 1. *A fork gate in a DNA walker circuit is deterministic if and only if there exists some guard G such that the left output track is guarded by G and the right is guarded by $\neg G$.*

Given any Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$, there exists a deterministic DNA walker circuit set S that can evaluate f, under any assignment to its n variables, such that $\texttt{TIME}(\mathsf{S}) = O(n)$. One construction is to simply form a canonical binary



(a) (b)

Fig. 3. (a) A connectivity graph of a DNA walker circuit to evaluate the disjunction $(X \vee Y \vee Z)$. There are two output tracks: one when the circuit evaluates to true, the other when it evaluates to false. The resulting path when $X = Y = f$, and $Z = t$ is shown highlighted. (b) Two copies of the disjunction circuit from (a) are composed into the conjunction $(A \vee B \vee C) \wedge (X \vee Y \vee Z)$. Two source nodes (two walkers) are used to evaluate clauses in parallel. No assignment of guards to the join gate labelled J can ensure that this circuit is deterministic. This is evident when $A = B = X = Y = f$ and $C = Z = t$.

decision tree over some fixed order of the n variables. However, in such a construction $\mathtt{SIZE}(\mathsf{S}) = \Theta(2^n)$. It is natural to consider more space efficient representations to evaluate f, such as binary decision diagrams (BDDs) [4]. In particular, reduced ordered BDDs are capable of representing some Boolean functions in a compressed form that can be exponentially smaller than its canonical binary decision tree representation. Like walker circuits, BDDs have a unique source. Unlike general BDDs, DNA walker circuits are necessarily planar. Either we are limited to considering planar BDD representations or additional fork and join nodes must be added to a BDD representation when realising it as a walker circuit. A significant difference, however, is that BDDs form directed acyclic graphs while tracks in a DNA walker circuit are undirected. Consider the case when a walker reaches a join gate via its left input track. Unless the right input track is blocked, the walker is equally likely to continue on the right input track as it is on the output track. Additional steps are necessary to compensate for the undirected nature of tracks in walker circuits.

Unlike fork gates, it is not obvious whether all join gates can be made deterministic. Theorem 2 characterizes both the necessary and sufficient conditions: a deterministic join of two disjoint sets of paths, one for each input track, is only possible if they were previously "forked"[4] on some variable X (*i.e.,* in one set all paths traverse an edge guarded by X and in the other set all traverse an edge guarded by $\neg X$). This property is exemplified by the contrast between the disjunction circuit of Fig. 3(a) and the conjunction of two disjunction circuits as shown in Fig. 3(b). In the conjunction, two walkers are used in an attempt to parallelize the evaluation. However, as the clauses do not have literals over a common variable, there are no guards that can be assigned to

[4] It is not a necessary condition that the two disjoint sets of paths reaching the join were forked by a common gate, only that they can be partitioned based on the value of some variable.

the join gate labeled J to ensure the circuit is deterministic. Note that this limitation is not caused by the restricted topology of walker circuits (*i.e.,* their layout on a planar surface), rather by the property that their tracks are undirected.

Theorem 2. *A join gate in a DNA walker circuit is deterministic if and only if there exists some guard G such that the left input track is guarded by G, the right by $\neg G$ and, prior to reaching those guards, all paths that can reach the left input must traverse a track guarded by G and all paths that can reach the right must traverse a track guarded by $\neg G$.*

2.3 Evaluating Boolean formulas with DNA walker circuits

Despite the shortcomings of join gates in current DNA walker circuits, it is not the case that Boolean formulas must be evaluated using a circuit forming a binary decision tree. Any Boolean formula can be represented in one of its canonical forms. We will focus on conjunctive normal form (CNF) which is a single conjunction of clauses, where each clause is a disjunction over literals. A formula in CNF is said to be k-CNF if the largest clause has size k. Using a standard transformation, a Boolean formula in k-CNF with at most l total literals can be converted to a 3-CNF formula over $O(l)$ variables, with at most $O(l)$ clauses (each having at most 3 literals). As such, we will reason exclusively about circuits to evaluate 3-CNF formulas.

Constructing a walker circuit to represent a formula in 3-CNF with m clauses is straightforward. Each clause can be represented by the disjunction circuit of Fig. 3(a). The source of the circuit will be the first fork gate of the first clause. The output track signalling the i-th clause is satisfied is connected to the input track of clause $i + 1$. Thus, the walker will only reach the single true sink of the circuit (output from clause m) if the formula is satisfied for that particular variable assignment. To ensure that both true and false signals can be reported deterministically, we use the reporting strategy depicted in Fig. 2 (Right) which employs two parallel copies of the circuit, each using different coloured walkers and different quenching sinks.

Theorem 3. *Let $\mathcal{F}$ be any 3-CNF Boolean formula with m clauses over n variables. There exists a DNA walker circuit set* S*, with* $\texttt{SIZE}(\mathsf{S}) = \Theta(m)$ *and* $\texttt{TIME}(\mathsf{S}) = O(m)$*, such that given any variable assignment A for $\mathcal{F}$,* $\texttt{VALUE}\,(\mathsf{S}[A])$ *is the truth value of $\mathcal{F}$ under assignment A.*

While the construction of Theorem 3 can represent any Boolean formula, and some in exponentially less space than a binary decision tree, the resulting circuit set is formula specific. Given the effort of creating DNA walker circuits, a more uniform circuit is worth exploring. As with silicon circuits, we can construct a uniform circuit to evaluate any 3-CNF formula, under any variable assignment, up to some bound on the number of variables. Each variable can be present in a clause as either a positive or negative literal, but not both. (The circuit can be modified to handle this case if necessary.) Therefore, there are at most $2^3\binom{n}{3}$ unique clauses in any 3-CNF Boolean formula over n variables, and also for any formula over $m \leq n$ variables. In this general circuit, we supplement each possible clause with an initial fork gate guarded on the condition of the clause being active or inactive in the particular formula being evaluated. If it is inactive, the walker can pass through to the output track denoting true, without traversing guards for the literals of the clause. Note that this only increases the size of each clause by a constant.

Corollary 1. *There exists a DNA walker circuit set* S*, with* $\texttt{SIZE}(\mathsf{S}) = O(n^3)$ *and* $\texttt{TIME}(\mathsf{S}) = O(n^3)$*, that can evaluate any* 3*-CNF Boolean formula over* $m \leq n$ *variables under any variable assignment.*

A 3-CNF formula with m clauses can be evaluated in polylogarithmic time (in m) using a silicon circuit in a straightforward manner: each clause can be evaluated in parallel and those results can be combined using a binary reduction tree of height $O(\log m)$—only if all clauses are satisfied will the root of the reduction tree output true. Is the same possible in DNA walker circuits? Unfortunately, this is not the case in general. Even with multiple walkers, the shortcomings of join gates can prevent the overall circuit from being deterministic. For example, consider if clause $m - 1$ and clause m have no literals over a common variable. By Theorem 2, a join gate connecting the circuits for these clauses cannot be deterministic. An example of this situation is given in Fig. 3(b).

3 Design and verification of DNA walker circuits

We have so far considered DNA walker circuits which work perfectly. However, in a real experimental scenario they are exposed to errors such as a blockade failing, or the walker crossing over to a different track. In this section, we analyse the reliability and performance of DNA walker circuits using probabilistic model checking. We develop a continuous-time Markov chain model, based on a variety of DNA walker experiments from [2, 13, 14], and analyse it against quantitative properties such as the probability of the computation finishing correctly. We use the PRISM model checker [8], which accepts models described in a textual language and properties in the form of temporal logic. A custom tool was developed to generate PRISM model scripts with matching track-layout graphs. Different configurations of tracks are studied: see Fig. 4 (top) for anchorage arrangements and Fig. 5 for track geometries. We use the results of experiments on linear (Fig. 4) and single-branched tracks to establish model parameters, and match model predictions with observations on double-layer tracks to evaluate the quality of our model.

Experiments show that the walker can step onto anchorages that are fixed as far away as $\sim$ 19 nm. We assume non-zero rates for the walker to step onto *any* intact anchorage within 24 nm distance. This range was chosen by taking into account the lengths of the empty anchorage and walker-anchorage complex, estimated at $\sim$ 15 nm and $\sim$ 11 nm respectively.

A step taken by the walker corresponds to a single transition in the Markov chain, although the real stepping process is more complex, as in Fig. 1. Assume that the *stepping rate* k depends on distance d between anchorages and some *base stepping rate* k_s. Denote by $d_a = 6.2$ nm the average distance between anchorages in the experiment shown in Fig. 4. Denote by $d_M = 24$ nm the *maximal interaction* distance discussed earlier. Based on previous experimental estimates of [14], we fix the stepping rate k as:

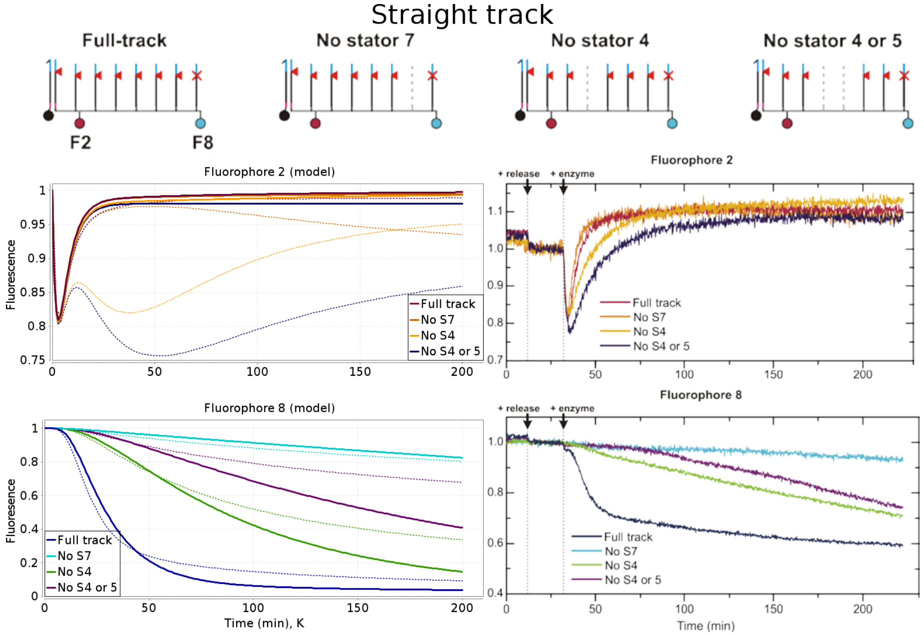


Fig. 4. Top: A small linear track of 8 anchorages with fluorophores on both the second and last anchorage. Experiments were performed with one or more anchorages omitted [14]. Right: Experimental results (reproduced with permission from the authors). The walker hardly reaches the final anchorage when anchorage 7 is removed, due to the double penalty of a longer final step *and* the mismatch in the final anchorage. Left: Model results. Dotted lines: Alternative model where the walker can step onto already-cut anchorages with rate $k_b = k_s/30$.

$$k = \begin{cases} k_s = 0.009\text{s}^{-1} & \text{when } d \leq 1.5d_a \\ k_s/50 & \text{when } 1.5d_a < d \leq 2.5d_a \\ k_s/100 & \text{when } 2.5d_a < d \leq d_M \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

These rates define a sphere of reach around the walker-anchorage complex, allowing the walker to step onto an uncut anchorage when it is nearby. In Fig. 5 the sphere of reach is depicted to scale with walker circuits. There are two exceptions. Stepping *from* the initial anchorage and stepping *onto* the final anchorage occur at lower rates. The domain complementary to the walker on the initial anchorage is two bases longer than the corresponding domain of a regular anchorage. Stepping from the initial anchorage was reported to happen 3× more slowly: this is incorporated in the model. The final anchorage includes a mismatched base that prevents cutting by the nicking enzyme. Based on the experimental data, we fit a tenfold reduction for the rate of stepping *onto* the final absorbing anchorage (Fig. 4).

Three types of interaction that are known to occur are omitted from the model: all three could be incorporated in future. Firstly, a rate of $k_s/5000$ is reported [14] for transfer of the walker between *separate* tracks built on different DNA origami tiles. Transfer between tracks could be eliminated by binding the tiles to a surface, thus keeping them apart. Secondly, the walker can move between intact anchorages in the absence of the nicking enzyme at a rate of $\sim k_s/13$ [14]. Enzymatic activity is relatively fast, with the cutting rate measured at $0.17s^{-1}$ while the enzyme binding to the DNA is considered not a rate limiting step [3]. We infer that the walker is attached to intact anchorages for only short periods of time. Thirdly, the walker can step backward onto cut anchorages. This requires a blunt-end strand-displacement reaction which is known to be slow relative to toehold-mediated displacement [16]. A variant of the model with a *backward* rate $k = k_s/30$ is shown in dotted lines in Fig. 4 (left). In this case the model predicts significant quenching of fluorophore F2 at late times by walkers whose forward motion is obstructed by omission of one or more anchorages: this does not match experimental data. A reduced rate $k_b = k/500$ (not plotted) has a similar effect.

The time-dependent responses of fluorescent probes F2 and F8 shown in Fig. 4 (left) are predicted by the Markov chain model using the rate parameters discussed above without any further fitting: they correspond well to the experimental data.

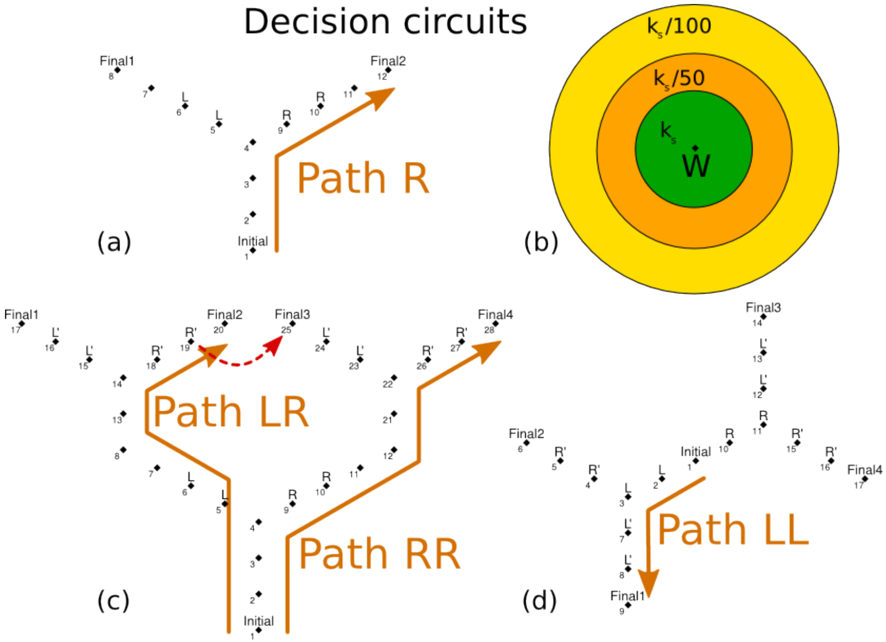
An additional parameter is needed to model branched tracks (Fig. 5 (a)). We introduce a *failure* rate for the anchorage blocking mechanism which is assumed to be the same for all junctions. We infer a failure rate of 30% by fitting to the results of the single-layer branched-track experiment illustrated in Fig. 5 [13].

3.1 Model results

Having used experiments on straight tracks and with a single layer of branching to determine the parameters of the model, we use the two-layer junction experiments shown in Fig. 5 (c) to evaluate its performance. In both the model and the experiments we can identify a difference between paths that follow the side of the track, namely paths LL and RR, and paths that enter the interior, paths RL and LR: the probability of a correct outcome for the side paths is greater. This is explained by *leakage transitions* between neighbouring paths, for example, see the red dotted line in Fig. 5. Walkers on an interior path can leak to both sides, but a path that follows the side can only leak to one side. This effect can also be shown by inspecting paths. By using a time-bounded 'until' property, we can reason about the probability of deviating from the intended path. For the double-layer track in Fig. 5, we infer that the probability of staying on the intended path *and* reaching the absorbing anchorage within 200 minutes is 55% for paths LR and RL, and 58% for paths LL and RR. The property used here is $P_{=?}[$ (correct-path) $U^{<200\text{min}}$ (finished-correct) $]$. This shows that walkers on interior paths are indeed more likely to deviate from the intended path than walkers on paths that follow the sides.

The double-layer track can be optimized by reducing the probability of leakage from the intended path. By decreasing the proximity of off-path anchorages and reducing the track length, both the proportion of walkers finishing and correctness are increased (see Fig. 5 (d)). The asymmetry between paths (LL, RR vs. LR, RL) also disappears.

Smaller tracks are not always better. In Fig. 6 several variants of a XOR-logic circuit are shown. The 'small', 'normal' and 'large' variants all use a total of four blocker strands per decision node. The large track is approximately as correct as the normal sized track, but a lower proportion of walkers reach an absorbing anchorage. The small track has a greater proportion of walkers that finish than the normal sized track, but it is considerably less correct.

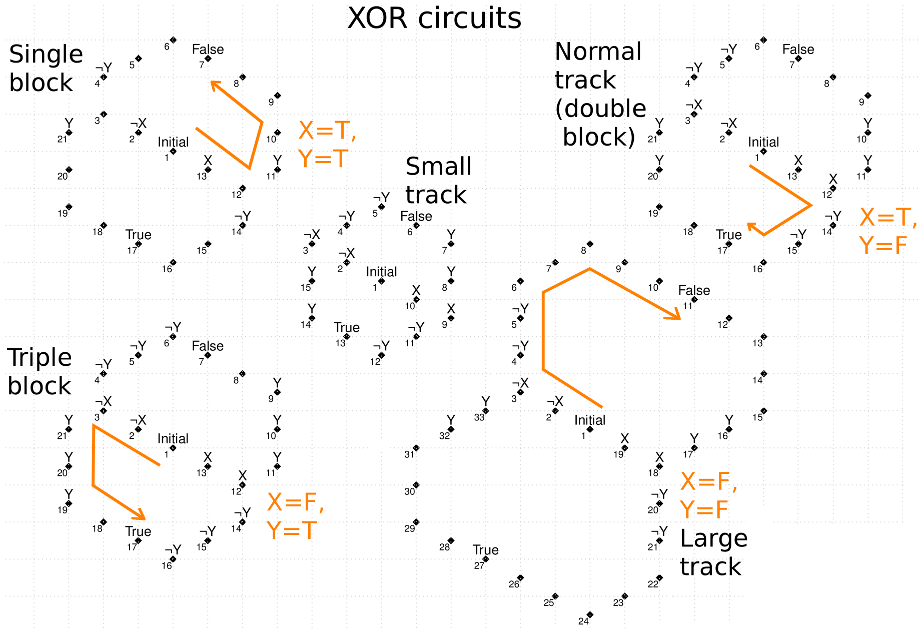


	Single layer			Single layer			2-Layer				2-Layer				Small 2-layer			
	Experiment			Model			Experiment				Model				Model			
%	R	R^2	L/R	R	R^2	L/R	RR	RL	LR	LL	RR	RL	LR	LL	RR	RL	LR	LL
Finishes	65	56	56	97	96	92	33	40	22	33	90	89	89	90	94	92	94	92
Correct	76	87	50	78	85	50	70	65	55	76	77	74	74	77	78	78	78	78
Deadlock				.081	.16	.0064					1.0	1.7	1.7	1.1	0.0	0.0	0.0	0.0
Steps				7.1	7.0	6.6					11.7	11.8	11.8	11.7	5.1	5.1	5.1	5.1

Fig. 5. Top: Track topology for single-layer (a) and double-layer (c,d) decision tracks. Initial indicates the initial anchorage, Final indicates absorbing anchorages, and L, L', R and R' indicate anchorages that can be blocked by input. Coloured circles (b) indicate the range of interaction of the walker. Bottom: Experimental results [13] compared with results from the model. Single layer track: R means a single blockade on the left, R^2 means a two-anchorage blockade on the left, L/R means single blockades on both the left and right. Double layer track: RL means anchorages labelled L and R' are blocked, so that the walker goes right on the first decision, and left on the second. All blockades are of two consecutive anchorages. All properties are given at time $T = 200$ min: "Finishes" is the probability that a walker quenches any fluorophore; "Correct" is the probability a finished walker quenches the correct fluorophore; "Deadlock" is the probability for the walker to get stuck prematurely, with no intact anchorage within reach; and "Steps" indicates the expected number of steps taken.

We infer that larger circuits are more susceptible to *deadlock*, based on Fig. 5 and 6. Deadlock occurs when a walker is isolated on a non-absorbing anchorage with no intact anchorage in range. From a computational standpoint deadlock is undesirable, as it is impossible to differentiate a deadlocked process from a live process.

The performance of PRISM depends on the model checking method. For small tracks as in Fig. 4, the property $P_{=?}[F^{[T,T]} \text{ position } = 2]$ for some time T is verified by PRISM with a precision of 10^{-6} within 10ms on common hardware [1]. Properties



%	Small track				Normal track				Large track				Single block				Triple block			
	TT	TF	FT	FF	TT	TF	FT	FF	TT	TF	FT	FF	TT	TF	FT	FF	TT	TF	FT	FF
Finishes	94	92	94	92	93	89	92	90	85	84	85	84	92	92	92	92	86	94	86	94
Correct	64	63	64	63	71	68	71	68	70	70	70	70	60	60	60	60	76	72	76	71
Deadlock	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.7	1.4	1.7	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Steps	5.8	5.9	5.8	5.9	7.8	7.6	7.8	7.6	11.0	11.6	11.0	11.6	8.9	8.5	8.9	8.5	7.7	6.9	7.7	6.9

Fig. 6. Performance analysis for a logic track expressing the XOR formula $(X \oplus Y)$. Properties verified are as in Fig. 5 at time $T = 200$ min.

for the single layer circuit in Fig. 5 were model checked within 3s to a precision of $< 10^{-6}$ using fast adaptive uniformisation [6]. For the dual-layer track in the same figure, single-threaded simulation of 10^5 paths yields a 95% confidence interval of size $< 0.4\%$ within 23 s [1].

4 Conclusions

Working from experimental observations, we have developed a simple model that explains the influence of track architecture, blockade failure and stepping characteristics on the reliability, performance and correctness of walker circuits. Model checking enables analysis of path properties and quantitative measures such as the expected number of steps, which cannot be established using traditional ODE frameworks. A major advantage of our approach is that circuit designs can be manipulated to study the properties of variant architectures.

We have also explored the potential for DNA walkers to evaluate Boolean functions. DNA walker ‘circuits’ can be designed to evaluate any Boolean function. We highlight that, in current experimental systems, paths within a circuit can only be joined under specific conditions. This severely limits the potential for parallel circuit evaluation using multiple walkers. This motivates, and could be addressed by, new design mechanisms capable of joining two arbitrary paths. Furthermore, current walker technology 'destroys' the track that is traversed. New mechanisms that can either replenish the track, or can avoid 'destroying' it, will lead to reusable circuits that have the potential for space efficient computation.

Acknowledgements We thank Jonathan Bath and Alex Lucas for useful discussion. The authors are supported by a Microsoft Research scholarship, ERC AdG VERIWARE, EPSRC Grant EP/G037930/1, a Royal Society - Wolfson Research Merit Award (AJT), and Oxford Martin School.

References

1. Intel i5-2520M, Fedora 3.8.4-102.fc17.x86_64, OpenJDK RE-1.7, PRISM 4.0.3
2. Bath, J., Green, S.J., Turberfield, A.J.: A free-running DNA motor powered by a nicking enzyme. Angewandte Chemie (International ed. in English) 44(28), 4358–61 (Jul 2005)
3. Bellamy, S.R.W., Milsom, S.E., Scott, D.J., Daniels, L.E., Wilson, G.G., Halford, S.E.: Cleavage of individual DNA strands by the different subunits of the heterodimeric restriction endonuclease BbvCI. Journal of molecular biology 348(3), 641–53 (May 2005)
4. Bryant, R.E.: Symbolic boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys 24(3), 293–318 (1992)
5. Chandran, H., Gopalkrishnan, N., Phillips, A., Reif, J.: Localized hybridization circuits. DNA Computing and Molecular Programming 6937, 64–83 (2011)
6. Dannenberg, F.G.W., Hahn, E.M., Kwiatkowska, M.: Computing cumulative rewards using fast adaptive uniformisation (Submitted)
7. Green, S., Bath, J., Turberfield, A.: Coordinated chemomechanical cycles: a mechanism for autonomous molecular motion. Physical review letters 101(23), 238101 (2008)
8. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. Computer Aided Verification 6806, 585–591 (2011)
9. Omabegho, T., Sha, R., Seeman, N.C.: A bipedal DNA Brownian motor with coordinated legs. Science 324(5923), 67–71 (2009)
10. Qian, L., Soloveichik, D., Winfree, E.: Efficient Turing-universal computation with DNA polymers. DNA Computing and Molecular Programming pp. 123–140 (2010)
11. Qian, L., Winfree, E.: Scaling up digital circuit computation with DNA strand displacement cascades. Science 332(6034), 1196–1201 (2011)
12. Seelig, G., Soloveichik, D., Zhang, D., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314(5805), 1585–1588 (2006)
13. Wickham, S.F.J., Bath, J., Katsuda, Y., Endo, M., Hidaka, K., Sugiyama, H., Turberfield, A.J.: A DNA-based molecular motor that can navigate a network of tracks. Nature nanotechnology 7(3), 169–73 (Mar 2012)
14. Wickham, S.F.J., Endo, M., Katsuda, Y., Hidaka, K., Bath, J., Sugiyama, H., Turberfield, A.J.: Direct observation of stepwise movement of a synthetic molecular transporter. Nature nanotechnology 6(3), 166–9 (Mar 2011)
15. Yin, P., Yan, H., Daniell, X.G., Turberfield, A.J., Reif, J.H.: A unidirectional DNA walker that moves autonomously along a track. Angewandte Chemie 43(37), 4906–11 (Sep 2004)
16. Zhang, D.Y., Winfree, E.: Control of DNA strand displacement kinetics using toehold exchange. Journal of the American Chemical Society 131(47), 17303–17314 (Dec 2009)

A Proofs of theorems

Proof (Proof of Theorem 1). If neither output track is guarded, then any path that can reach the gate could be extended via the left or the right output track and the gate would not be deterministic. Similarly, this is true when only one output track is guarded and the guard evaluates to false. (If the fork gate is only reachable when the single output guard evaluates to true, then the gate is redundant as the output track with the guard is never used.) Thus, consider when each output track is guarded and let the left output have guard G_L and the right have guard G_R. Note that $G_L \not\equiv G_R$ as otherwise any path that reaches the gate will result either in a deadlock — when both evaluate to false — or the path could be extended via the left or the right output tracks — when both evaluate to true. Consider any path p that can reach the gate and the case when G_L evaluates to true and G_R to false. It follows that, before reaching the gate, p must not traverse a track guarded by $\neg G_L$ nor by G_R. Since the gate is non-redundant, p must also be able to reach the gate when G_L evaluates to false and G_R to true. It follows that, before reaching the gate, p must not traverse a track guarded by G_L nor by $\neg G_R$. Therefore, path p is independent of the variables affecting guards G_L and G_R. Thus, there exists a variable assignment such that any path reaching the gate will result in a deadlock, or can be extended via both output tracks, unless $G_L \equiv \neg G_R$. □

Proof (Proof of Theorem 2). (⇒ if) Suppose the left input track is guarded by G, the right by $\neg G$ and, prior to reaching those guards, all paths that can reach the left input must traverse a track guarded by G and all paths that can reach the right must traverse a track guarded by $\neg G$. There are two cases to consider. Suppose G evaluates to true. Then, no path can reach the right input since, by the assumption, those paths must traverse a track guarded by $\neg G$ prior to reaching the gate. It follows that all paths that can reach the gate when G evaluates to true must be to the left input. Furthermore, as the right input is guarded by $\neg G$, those paths can only be extended via the output of the gate. The other case (G evaluates to false) is symmetric. Furthermore, as the guards are negations of each other, they cannot simultaneously evaluate to false and cause a potential deadlock.

(⇐ only if) Let G_L and G_R be the guards of the left and right inputs, respectively. (If one or more of the input tracks is unguarded, then the gate cannot be deterministic when both are reachable by at least one path.) First, consider all paths that can reach the left input, guarded by G_L. It must simultaneously be true that none of those paths (i) traverse a track guarded by $\neg G_L$ and (ii) all of those paths traverse a track guarded by $\neg G_R$. If condition (i) is not satisfied, then there would exist a path that traverses a track guarded by $\neg G_L$ and, to extend past the join gate, must traverse another guarded by G_L. As this is not possible, the path would end in a deadlock and the gate would not be deterministic. If condition (ii) is not satisfied then there would exist some path p that does not traverse a track guarded by $\neg G_R$, but may possibly traverse a track guarded by G_R. In this case, there exists a variable assignment where G_R, and all other guards on path p, evaluate to true. With such a variable assignment, path p could be extended via the output track or the right input track. Thus, condition (ii) must also be satisfied otherwise the gate would not be deterministic. The conditions (and the argument that both are necessary) when considering all paths that can initially reach the right input, guarded by G_R, are symmetric.

The sufficiency argument ($\Rightarrow$ *if*) shows the gate is deterministic when $G_L \equiv \neg G_R$. It remains to show it is not deterministic otherwise. First, consider the consequence when both G_L and G_R evaluate to true. By condition (ii) all paths leading to the left (right) input traverse a track guarded by $\neg G_R$ ($\neg G_L$). In this case, no paths can reach the gate. Recall that the gate is non-trivial and therefore each input is reachable by at least one path. Thus, consider when both G_L and G_R evaluate to false. The conditions permit that paths can reach the gate, however, if any path does it will deadlock as both inputs to the gate are blocked. Thus, for all paths that can reach the gate, it will be deterministic only when $G_L \equiv \neg G_R$. □

Proof (Proof of Theorem 3). The construction is described in Section 2.3 and it is easy to see that the circuit is deterministic and that it correctly reports the truth value of $\mathcal{F}$ under assignment A. What remains is to bound the circuit size and worst case time. The construction uses a set of two circuits: $\mathcal{C}_T$ and $\mathcal{C}_F$. Consider the circuit $\mathcal{C}_T$ used to evaluate if $\mathcal{F}$ is true under assignment A. There are m clauses and each is simulated by a disjunction circuit of size $O(1)$. These circuits are composed in series to form $\mathcal{C}_T$. Therefore, $\texttt{SIZE}(\mathcal{C}_T) = \Theta(m)$ and $\texttt{TIME}(\mathcal{C}_T) = O(m)$. The arguments are the same for circuit $\mathcal{C}_F$ and, as both are evaluated in parallel, the claim follows. □

B Composable DNA walker circuits

A DNA walker circuit can realize a number of different Boolean functions by interpreting different blocking anchorages as different Boolean guards. Figure 7 shows how the same circuit topology can realize eight different Boolean functions, over two variables. Furthermore, each instantiation is fully composable: its source can be connected to the output of a previous circuit, and both its true and false outputs can be connected to other circuits.

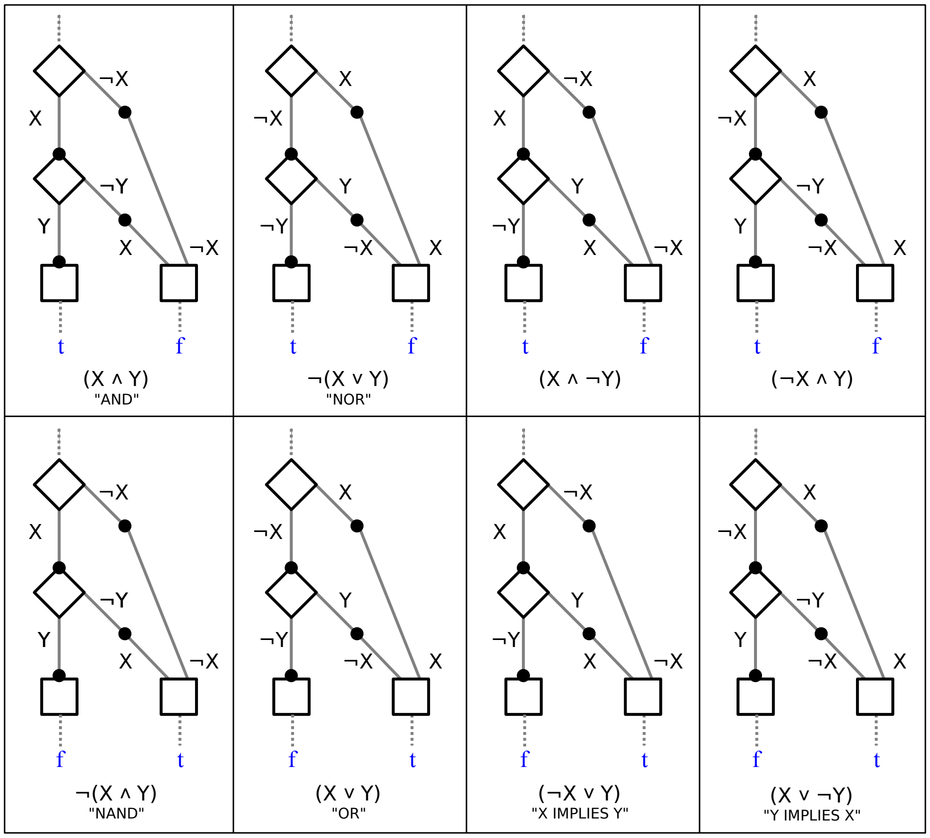


Fig. 7. The same DNA walker circuit topology shown for four different Boolean guard assignments and two different labellings of output tracks. Each realize one of eight different Boolean functions over two variables and are also composable with other circuits.