BioSystems 109 (2012) 91–104

Contents lists available at SciVerse ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

DNA biosensors that reason

Iñaki Sainz de Murieta*, Alfonso Rodríguez-Patón**

Departmento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo, s/n, E-28660 Boadilla del Monte, Madrid, Spain

ARTICLE INFO

Article history:
Received 13 August 2011
Received in revised form 23 January 2012
Accepted 16 February 2012

Keywords:
DNA computing
Inference
Biomolecular computation
Strand displacement
Biosensor
In vitro diagnosis

ABSTRACT

Despite the many designs of devices operating with the DNA strand displacement, surprisingly none is explicitly devoted to the implementation of logical deductions. The present article introduces a new model of biosensor device that uses nucleic acid strands to encode simple rules such as "IF DNA_strand_1 is present THEN $disease_A$" or "IF DNA_strand_1 AND DNA_strand_2 are present THEN $disease_B$". Taking advantage of the strand displacement operation, our model makes these simple rules interact with input signals (either DNA or any type of RNA) to generate an output signal (in the form of nucleotide strands). This output signal represents a diagnosis, which either can be measured using FRET techniques, cascaded as the input of another logical deduction with different rules, or even be a drug that is administered in response to a set of symptoms. The encoding introduces an implicit error cancellation mechanism, which increases the system scalability enabling longer inference cascades with a bounded and controllable signal–noise relation. It also allows the same rule to be used in forward inference or backward inference, providing the option of validly outputting negated propositions (e.g. "diagnosis A excluded"). The models presented in this paper can be used to implement smart logical DNA devices that perform genetic diagnosis in vitro.

1. Introduction

Competitive hybridization between DNA strands is the foundation of a process called strand displacement. This method is used in biomolecular computation to implement computing operations (Reif and Sahu, 2009; Takahashi et al., 2006; Rinaudo et al., 2007; Macdonald et al., 2008; Smolke, 2009; Soloveichik et al., 2008; Cardelli, 2009; Cockroft and Ghadiri, 2007; Wasiewicz et al., 2001; Zhang et al., 2007; Seelig et al., 2006). In short, it can be defined as follows: a strand A displaces another strand B from a complex $A'B$, due to the higher affinity between A and A' and the greater stability of the duplex AA'.

Despite the many designs of DNA devices operating with strand displacement, surprisingly none is explicitly devoted to the implementation of logical deductions, which, in mathematics and logic, is called inference: given a set of facts (e.g. "DNA_strand_1 AND DNA_strand_2 are present") and a set of implication rules (e.g. "IF DNA_strand_1 AND DNA_strand_2 are present THEN $disease_A$"), the inference process derives new facts as conclusions of the original set of implications and facts (the conclusion is "$disease_A$").

Our interest in performing inference with DNA strands is mainly inspired by the previous work of Ran et al. (2009). They developed an autonomous molecular system that, encoding facts and rules as DNA strands and using restriction enzymes as an inference engine, was able to perform simple logical deductions. Our goal is the same, but using different biological hardware. Instead of using restriction enzymes, we take the DNA strand displacement as the main biological operation. With this motivation, we already presented an initial version (Rodríguez-Patón et al., 2010) implementing logical inference with DNA strands using strand displacement.

2. Materials and methods

2.1. Previous work

Biomolecular computation is a discipline that merges computer science and biotechnology, using biomolecules as information processing substrate. Leonard Adleman's seminal work (Adleman, 1994) emerged as the proof of concept of this new discipline. Adleman was the first to solve difficult computing problems, such as the Hamiltonian path problem (finding a path in a graph that visits each vertex exactly once), taking advantage of the parallel processing capabilities of recombinant DNA. Subsequently Richard Lipton was equally successful at tackling another difficult mathematical problem (Lipton, 1995), using the same set of basic DNA operations: synthesis, amplification, append, extraction, detection and polymerization. By applying this set of operations on a population of DNA strands, encoding all the potential solutions of the problem, they were able to filter out only the right ones.

During the early years of this discipline, the problems approached were mainly combinatorial, but, in 2006, there

* Corresponding author.
** Principal corresponding author. Tel.: +34 91 336 6604.
E-mail address: arpaton@fi.upm.es (A. Rodríguez-Patón).

0303-2647/$ – see front matter © 2012 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2012.02.005

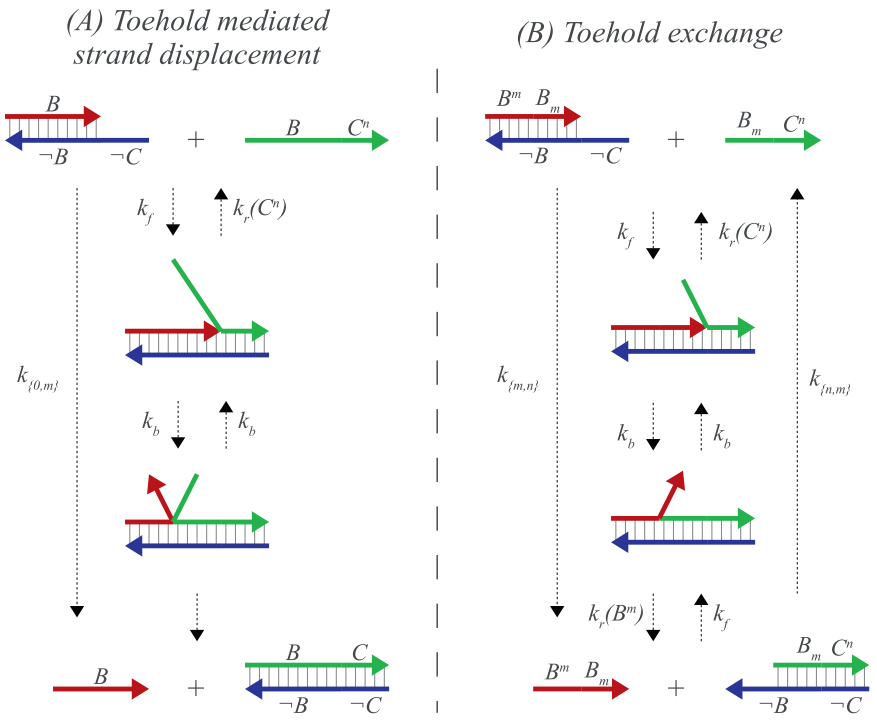

Fig. 1. Toehold mediated strand displacement and toehold exchange. Capital letters represent nucleotide domains, and the negation symbol ¬ denotes the Watson–Crick complement of the domain (if P is a generic nucleotide domain, $\neg P$ represents its Watson–Crick complement). The use of superscripts and subscripts within a domain P denotes the 5′ and 3′ portions of the domain, respectively. A superscript value of m indicates the 5′-most m bases of the full domain, whereas a subscript value of m indicates all but the 5′-most m bases of the domain. Thus the concatenation of P^m and P_m form the full domain P for all values of m. (A) *Toehold mediated strand displacement*. Two molecules are illustrated at the top of the figure: (1) a double stranded complex, with the domain B as the upper strand and the concatenation of domains $\neg B$ and $\neg C$ as the lower strand, and (2) a single strand concatenating the domains B and C. As the domains C and $\neg C$ are complementary, they bind to each other with a forward hybridization rate k_f; the length of the toehold C, n, is short enough to consider this reaction can be reversed with a dissociation rate $k_r(C^n)$, n being the length of domain C. After C and $\neg C$ are bound, the resulting molecule starts a process called *branch migration*, where the two domains with nucleotide sequence B compete to bind to domain $\neg B$ in the lower strand, moving the border between both upper strands left and right. After some point, the strand B will be released from the complex, letting BC and $\neg B \neg C$ form a double stranded complex whose length is long enough to consider the reaction irreversible. (B) *Toehold exchange*. This method is similar to toehold mediated strand displacement in that a single strand binds by a toehold to a double strand to initiate a branch migration. However, they differ in the length of the initial single strand (invading strand), which is shorter than its counterpart on side A of this figure. In consequence, after the toeholds C and $\neg C$ bind to each other reversibly, the branch migration movements are not enough to achieve a complete displacement of strand B: the incumbent toehold, B^m, is still bound to the lower strand of the complex. The process will finish when B^m spontaneously dissociates from the complex at rate $k_r(B^m)$. Contrary to (A), this last step is reversible since the new double stranded complex has a new toehold, which can bind again to B^m with a hybridization rate k_f. The *bimolecular reaction model (BM)* (Zhang et al., 2007) of toehold exchange approximates the constant rate of the overall reactions depicted by the long arrows at the sides in (B), obviating the need to calculate all the intermediate reaction rates (k_f, $k_r(B^m)$, $k_r(C^n)$, . . .). It is denoted as $k_{\{m,n\}}$, meaning that the lengths of invading toehold (m) and incumbent toehold (n) are all that is needed for its calculation. It can also be applied in toehold mediated strand displacement reactions, assuming an incumbent toehold of length zero (see side A of the figure).

emerged another research line in DNA computing based on the competitive hybridization of DNA: the DNA strand displacement operation. Since then, several contributions have been made in this direction, like, for example, the design of logic gates (Seelig et al., 2006; Cockroft and Ghadiri, 2007), general purpose catalytic gates (Qian and Winfree, 2009, 2011a,b; Qian et al., 2011), DNA automata (Takahashi et al., 2006), as well as theoretical models (Cardelli, 2009).

The strand displacement operation can be used with any type of nucleic acid, like, for example, the different RNA molecules involved in the RNA interference process. Research presented in Xie et al. (2010) approached the usage of synthetic RNA-based biosensors that, by means of the strand displacement operation, transduce input mRNA levels into small interfering RNA (siRNA) that suppress the translation of specific target RNA molecules.

There are some precedents on the use of DNA molecules to perform logical inference, such as the use of self-assembly and polymerase chain reaction (PCR) to build boolean functions (Wasiewicz et al., 2001) and perform inference (Wasiewicz et al., 2000), the molecular representation of formulas and querying using DNA hairpins (Hagiya et al., 1997; Rose et al., 2006), as well as the implementation of deduction with simple rules (Kobayashi, 1999).

The work presented in Ran et al. (2009) reawakened the interest and potential of applying the logic programming paradigm in biomolecular computation. Based on the DNA automaton concept developed in previous research (Benenson et al., 2004, 2003, 2001), they built a system capable of performing simple logical deductions with DNA molecules. In that work, propositions and implication rules were encoded using double stranded DNA molecules with a free sticky end; when a proposition and an implication had complementary sticky ends, both molecules merged into one; then the enzyme Fok I would cleave the resultant molecule into two new pieces, different from the original fact and implication molecules; one of these new pieces, merged with an auxiliary DNA strand, would represent the conclusion inferred from fact and implication, which could either be cascaded into another inference process or be read as output using FRET techniques (see Fig. 2).

Motivated by Ran et al. (2009), our group presented a system with similar functional power elsewhere (Rodríguez-Patón et al., 2010). It differed as follows:

- Instead of restriction enzymes, we used the DNA strand displacement operation as a biological engine.
- Whereas Shapiro's model represented the logical value "False" as the absence of DNA strands, our model encoded it using the Watson–Crick complementary strand of the corresponding

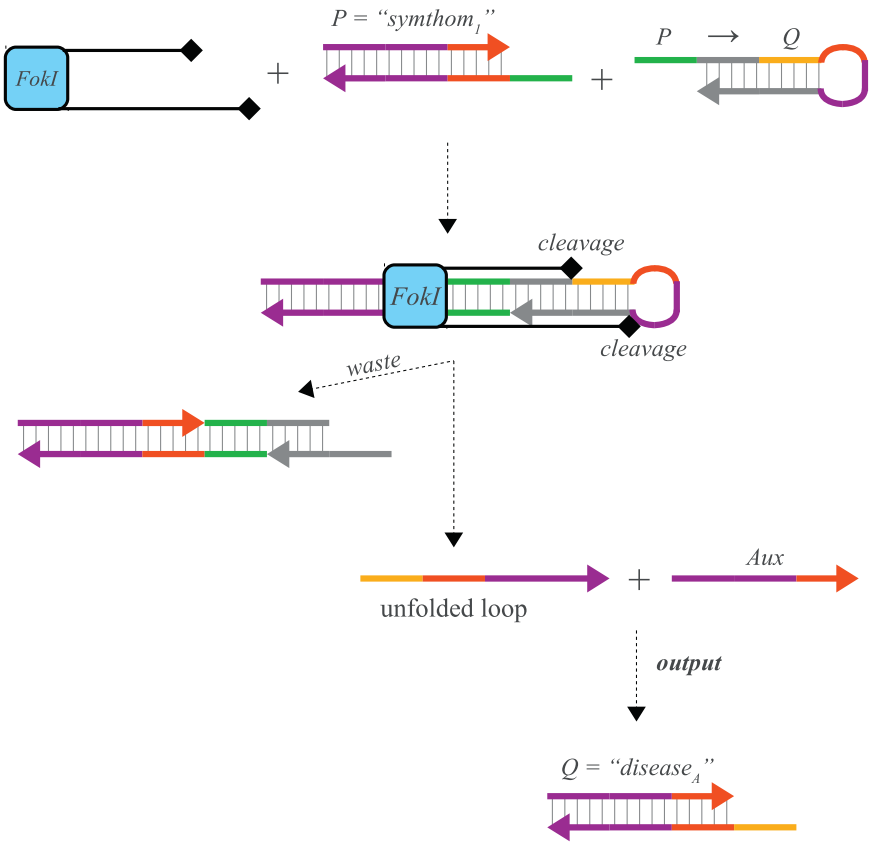

Fig. 2. Molecular implementation of simple logic programs. The figure represents the basic operation of the DNA inference machine developed in Ran et al. (2009). Propositions are encoded in double stranded molecules with a sticky end. The double stranded area includes the recognition sequence of the enzyme FokI. The sticky end nucleotide sequence uniquely encodes an input proposition (in the figure, P = "$symptom_1$").

proposition with the logical value "True", as suggested in Sakamoto et al. (2000).

- The very same implication device could trigger an inference process in the presence of the premise (applying modus ponens) or in presence of the negated consequence (applying modus tollens).

We continued studying the strand displacement operation (Rodríguez-Patón et al., 2011), presenting a computing model which performs a chain of logical resolutions with logical formulae in conjunctive normal form.

This article builds on our previous work, with the aim of modeling of sensors that can be applied to *in vitro* diagnosis processes.

2.2. Inference and logic programming

A logic program (Lloyd, 1987; Sterling and Shapiro, 1994) is a finite set of facts and implications that, starting from a set of premises (axioms or initial input conditions) and following a set of deduction rules (inference rules), lead to one or more conclusions (outputs or logical consequences). This reasoning process, also called inference, is the engine of logic programming.

When referring to facts we will use the term "proposition", which is the minimal syntactic entity in propositional logic. Propositions refer to simple affirmative statements, and we may represent any given proposition with a letter, like, for example, P = "DNA strand present" or Q = "disease present". A proposition can take the values "True" or "False". P = "True" is represented as P, whereas P = "False" is represented as $\neg P$.

Implications establish relations between input propositions (the premise) and output propositions (the conclusion). They can be represented as "IF–THEN" rules. For example, the statement "IF P THEN Q" is an implication, which means that if P is present, then Q must also be present. In logic, implications are represented with an arrow that goes from the premise to the conclusion; therefore we can rewrite the above implication as $P \rightarrow Q$.

Premises and conclusions can be made of more than one fact. For example, the implication "IF P AND Q THEN R" means that, if both input propositions P and Q are true, then R must also be considered true. In logic, the connective AND is represented with the symbol $\wedge$, so the previous implication can be rewritten as $P \wedge Q \rightarrow R$. Another example can be taken from the implication "IF P THEN Q OR R": it means that, if the input proposition P is true, then either Q or R (or both) must be true too. In logic, the connective OR is denoted by the symbol $\vee$; therefore we can rewrite the previous implication as $P \rightarrow Q \vee R$.

The two basic inference rules are called modus ponens and modus tollens. The deduction rule modus ponens states that, from a fact P and an implication $P \rightarrow Q$, one can deduce Q. It can be formally expressed as follows: $(P \rightarrow Q) \wedge P \rightarrow Q$. The deduction rule modus tollens states that, from $\neg Q$ and the implication $P \rightarrow Q$, one can deduce $\neg P$. It can be formally expressed like this: $(P \rightarrow Q) \wedge \neg Q \rightarrow \neg P$. For example, let the implication $P \rightarrow Q$ be our logic program. $P \rightarrow Q$ represents the statement "IF DNA strand present THEN disease present" (or DNA strand $\rightarrow$ disease). If we add the proposition P = "DNA strand present" as the program input, the modus ponens inference rule is triggered, deducing the proposition Q = "disease present" as the output. On the contrary, if we add the proposition $\neg Q$ = "disease not present" as the program input, the

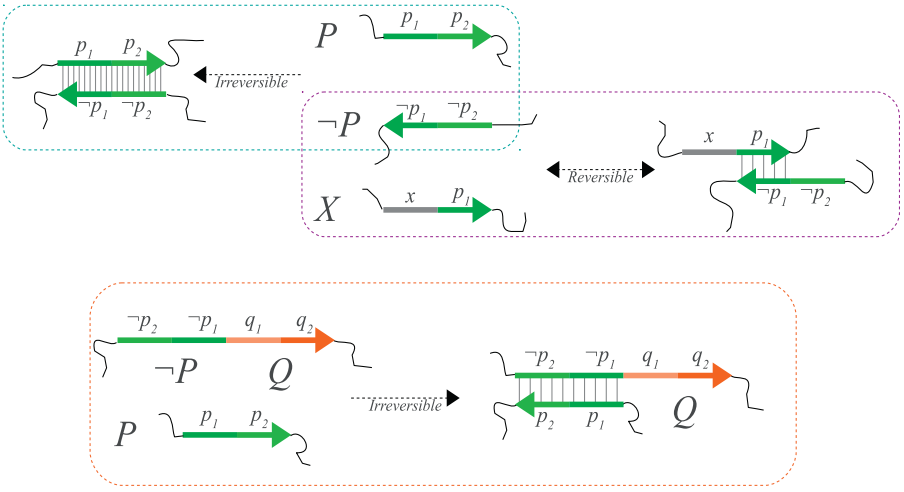

Fig. 3. Encoding of propositions. Propositions are encoded to comprise two consecutive domains in a nucleotide strand (see strand P, comprising domains p_1 and p_2, and strand $\neg P$, comprising domains $\neg p_1$ and $\neg p_2$). The right-hand reaction shows a hybridization between two strands, X and $\neg P$, where only domain p_1 in X and $\neg p_1$ in P are complementary; such a bond with the length of one domain is unstable enough to consider the reaction reversible. On the contrary, the left-hand reaction shows a hybridization between two strands, P and $\neg P$, where both domains are totally complementary: p_1p_2 in P to $\neg p_1 \neg p_2$ in $\neg P$; this bond comprising two domains is, due to its longer length, stable enough to consider the reaction irreversible. The reaction at the bottom, will be irreversible as long as no strand comprising the domains $\neg q_1 p_1$ performs a strand displacement reaction.

modus tollens inference rule is triggered, deducing the proposition $\neg P$ = "DNA strand not present" as output.

2.3. DNA base-pairing and strand displacement

DNA is the molecule that encodes the information of life. It occurs naturally as two strands forming a double helix. Each strand is a long polymer formed by a sequence of nucleotides, each consisting of a sugar, a phosphate group and a base. The union between both strands is possible thanks to Watson–Crick complementarity, which establishes a specific interaction between complementary bases: adenine (A) always binds to thymine (T), whereas cytosine (C) binds to guanine (C) (in RNA, the base uracil (U) replaces thymine). DNA strands have two different ends: one is named 3′ and the other 5′. When two DNA strands are complementary, they couple in an antiparallel fashion, meaning that the 3′ end of the first strand binds to the 5′ end of the second strand, and vice versa.

DNA strand displacement was proposed in 2006 as a mechanism for performing computation with DNA strands (Seelig et al., 2006). The basic mechanism of strand displacement is denoted *toehold mediated strand displacement*: if we have a DNA duplex PQ consisting on strands P and Q, where P is not fully complementary to P, in the presence of a strand P' totally complementary to P, the system evolves its configuration from PQ to a DNA duplex PP' consisting on strands P and P', releasing the strand Q from its original duplex. Fig. 1(A) provides a detailed description of this strand displacement variant. Toehold mediated strand displacement will explain most of the strand displacement reactions shown in this article.

As an evolution of strand displacement, the *toehold exchange* mechanism was introduced to achieve a better control of the strand displacement dynamics (Zhang et al., 2007) (see Fig. 1(B)).

2.4. Simulation tools

The DNA devices presented in these paper have been modeled and simulated using Cain, version 1.4. Cain performs stochastic and deterministic simulations of chemical reactions. If offers a wide variety of solvers, including Gillespie's methods and Ordinary Differential Equations (ODE) integration. The reactions in our devices have been modeled according to mass action kinetic laws, and simulated with an ODE solver: Runge–Kutta (Cash–Karp), with 1000 frames, 32 bins, histogram multiplicity 4 and error rate 10^{-15}.

The values for the kinetic constants have been calculated using the following approximation (Zhang et al., 2007): $k_f = 3.5 * 10^6\,\mathrm{M}^{-1}\,\mathrm{s}^{-1}$, $k_r = k_f * 2/x * e^{\Delta G^\circ(\delta)/RT}\,\mathrm{s}^{-1}$ and $k_b = 400/x^2\,\mathrm{s}^{-1}$, at 25 °C and 11.5 mM Mg^2 or 1 M Na^+. Assuming all the toehold domains in our models are 6 base pairs long ($x = 6$ in the formulas above) and have an average ΔG° of −10 kcal/mol, the following rates are obtained and used in our simulations: $k_f = 3.5 * 10^6\,\mathrm{M}^{-1}\,\mathrm{s}^{-1}$, $k_r = 5.35 * 10^{-2}\,\mathrm{s}^{-1}$ and $k_b = 11.1\,\mathrm{s}^{-1}$. Rates used in Section 3.5 (Error measurement) have been directly taken from the rough estimates presented in Zhang et al. (2007).

3. Results

3.1. Encoding of propositions

The model proposed in this paper encodes propositions and their truth values with DNA strands. Given a generic proposition P, it is assigned a strand which comprises a unique nucleotide sequence. That sequence is divided into two domains of equal length, p_1 and p_2, oriented in a way that the domain with subscript 1 is closer to the 5′ end, whereas the domain with subscript 2 is closer to the 3′ end. The negation of the above generic proposition, denoted as $\neg P$, is represented by assigning it a strand comprising the Watson–Crick complementary sequence of P. $\neg P$ is also divided into two domains of equal length, $\neg p_1$ and $\neg p_2$, oriented in a way that the domain with subscript 1 is closer to the 3′ end, whereas the domain with subscript 2 is closer to the 5′ end (see Fig. 3).

Each domain p_i is considered a toehold, and can reversibly bind to a complementary toehold as depicted in the reaction between X and $\neg P$ in Fig. 3. However, when a strand comprising at least two consecutive toehold domains binds with its complementary strand, we can consider it an irreversible reaction, since the longer length of the resultant double strand significantly lowers the dissociation rate of the complex. This is illustrated in Fig. 3 between P and $\neg P$.

3.2. Encoding of implications

Having set up the proposition encoding model, the next step is to encode implications (or simple rules), with one premise and one conclusion: $P \rightarrow Q$. The center panel in Fig. 4 illustrates the design of this basic implication.

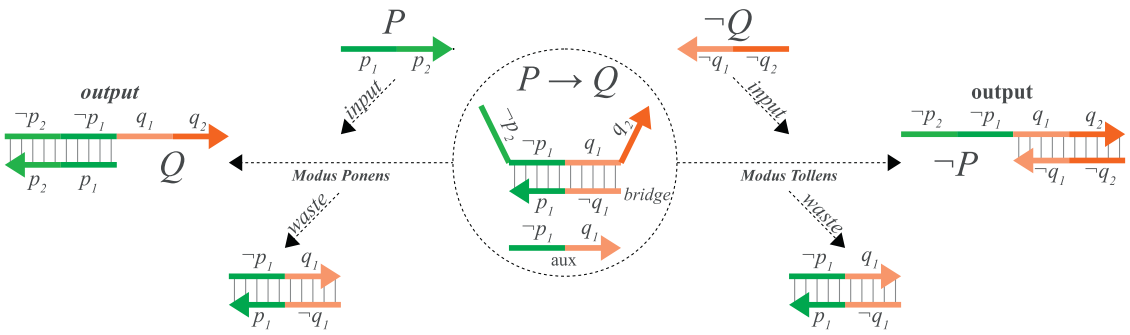

Fig. 4. Basic inference step. The left side of the figure shows the application of the modus ponens inference rule when an input P interacts with the implication $P \rightarrow Q$: as the input domain p_2 is complementary to one of the free domains in $P \rightarrow Q$, a strand displacement reaction binds both domains of P to the premise domains of the implication, unbinding p_1 in the bridge strand. At this point there is only one domain linking the bridge strand to the premise and conclusion, so, either spontaneously or with the help of the auxiliary strand, it will dissociate from the upper strand forming a duplex structure (waste). The remaining strands form the output, composed of a "history" section (the double stranded region) and an active section (the single stranded region) that encodes the output proposition Q. The right part of the figure shows the application of the modus tollens inference rule when an input $\neg Q$ interacts with the implication $P \rightarrow Q$: as the input domain $\neg q_2$ is complementary to one of the free domains in $P \rightarrow Q$, a strand displacement reaction binds both domains of $\neg Q$ to the conclusion domains of the implication, unbinding $\neg q_1$ in the bridge strand. At this point there is only one domain linking the bridge strand to the premise and conclusion, so, either spontaneously or with the help of the auxiliary strand, it will dissociate from the upper strand forming a duplex structure (waste). The remaining strands form the output, composed of a "history" section (the double stranded region) and an active section (the single stranded region) that encodes the output proposition $\neg P$.

The left panel in Fig. 4 shows the application of the modus ponens inference rule: the proposition P is input. As one of its domains (p_2) is complementary to the free domain ($\neg p_2$) in $P \rightarrow Q$, a strand displacement reaction binds both domains of P to the premise domains of the implication, unbinding p_1 in the bridge strand. At this point there is only one domain ($\neg q_1$) linking the bridge strand to the premise and conclusion, so, either spontaneously or with the help of the auxiliary strand, it will dissociate from the upper strand forming a duplex structure (waste). The remaining strands conform the output, composed of a "history" section (the double stranded region) and an active section (the single stranded region) that encodes the output proposition Q. The right panel in Fig. 4 shows how the modus tollens inference rule is symmetrically applied when the proposition $\neg Q$ is input, releasing $\neg P$ as output.

This device may remind the seesaw DNA gate motif originally presented in Qian and Winfree (2009) and further developed recently (Qian and Winfree, 2011a,b; Qian et al., 2011). But they clearly differ in the fact that our DNA motif is not catalytic. In appearance, our motif is more similar to one of the previous models presented by our group in Rodríguez-Patón et al. (2010). It differs as to:

1. The single stranded domains of the implication, $\neg p_i$ and q_i, behave as toeholds, and thus cannot form a stable duplex with a complementary strand of the same length. For a duplex to be stable, it should be composed of at least two consecutive domains on each single strand of the complex.
2. The potential output propositions, $\neg P$ and Q, are concatenated in the same DNA strand, unlike in Rodríguez-Patón et al. (2010) where they are encoded in different strands. This increases the stability of the duplex formed by $p_1\, \neg p_1$ and the bridge strand, keeping two domains per strand.

Inputs and outputs do not appear to have the same format, but, for our purpose, they are equivalent: as long as the output contains a section with two consecutive domains formatted as a proposition, it will be possible to reuse it as an input for another implication. Fig. 5 shows an example: the proposition P is input to a system containing the rules $P \rightarrow Q$ and $Q \rightarrow R$, triggering a chain reaction that releases R as output. Note how a history of the previous inference steps is accumulated in the double stranded section of the output. This is an improvement on the methods proposed in Rodríguez-Patón et al. (2010), as none of those keeps a log of the previous inference steps.

3.2.1. Kinetic model and simulation

The design of the implication device described above will now be modeled according to mass action kinetic laws. The list of equations below translates the modus ponens inference rule described in Fig. 4 (modus tollens would work symmetrically). It contents not only the main reaction flow that drive the steps described there, but also the side reactions representing the crosstalk from the rest of potential reactions between the input molecules, the device molecules and the potential intermediate molecules (see Fig. 6 for a graphical representation):

Main reaction flow:

$$P + PQ : bridge \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : PQ : bridge_1$$

$$P : PQ : bridge_1 \underset{k_b}{\overset{k_b}{\rightleftharpoons}} P : PQ : bridge_2$$

$$P : PQ : bridge_2 + aux \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : PQ : bridge : aux_2$$

$$P : PQ : bridge : aux_2 \overset{k_b}{\rightarrow} P : PQ + bridge : aux$$

Side reactions:

$$P + aux \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : aux$$

$$P : aux + PQ : bridge \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : PQ : bridge : aux_1$$

$$aux + P : PQ : bridge_1 \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : PQ : bridge : aux_1$$

$$bridge + aux \overset{k_f}{\rightarrow} bridge : aux$$

Fig. 7 shows the results of simulating the above equations in Cain with the parameters specified in Section 2.4 and two different initial concentrations of input (P) and implication devices ($PQ : bridge$ and aux). In Fig. 7A, all the initial concentrations were set to 1 nM. We can see how the concentration growth of $P : PQ$ and $bridge : aux$ increases exponentially at the beginning, but becomes flatter as the concentrations of P, $PQ : bridge$ and aux decrease. Concentration of intermediate molecules is always kept to minimum values. It can be easily seen in the plot that the system would need around 8.3 h (30,000 s) to produce 70% of the input concentration. A potential way to accelerate the process, intuited from the system's behavior

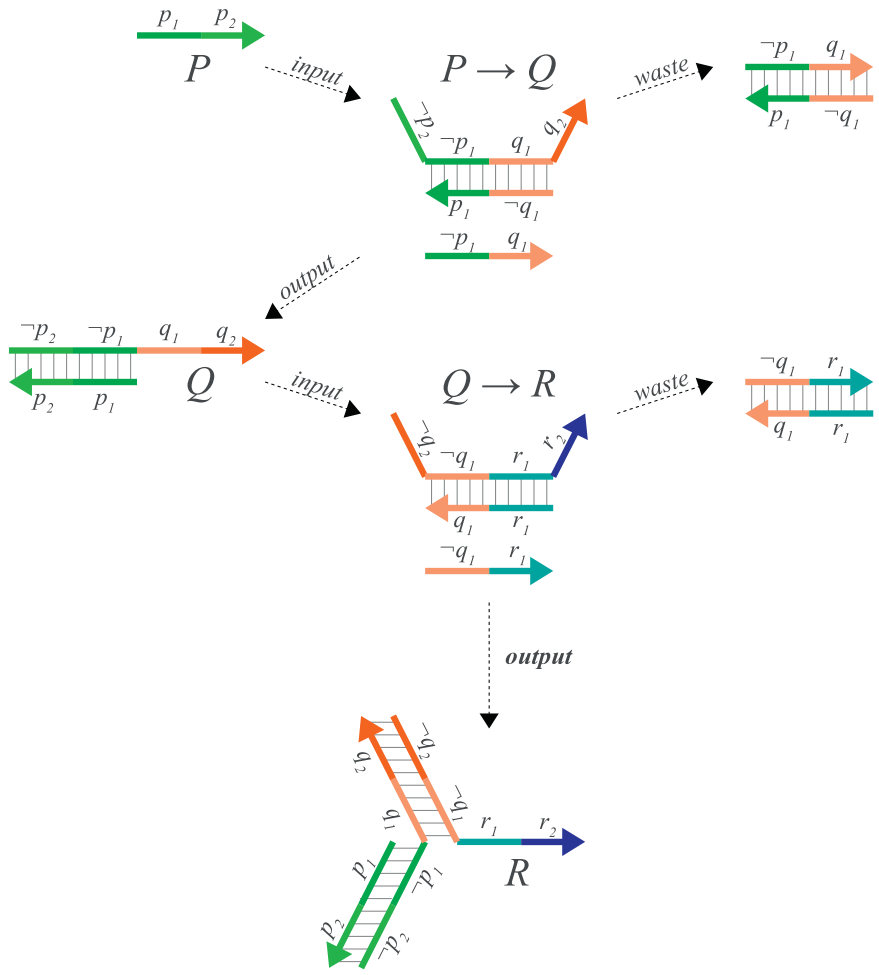

Fig. 5. Iterated modus ponens. At the beginning of the process, the systems contains only the implications $P \rightarrow Q$ and $Q \rightarrow R$. The top part of the figure shows how the implication $P \rightarrow Q$ releases the output Q when P is input. At the center we see how this intermediate output Q is taken as input by the implication $Q \rightarrow R$, releasing a molecule that comprises the final output R.

in Fig. 7A, would be to increase the concentration of device molecules ($P:PQ:bridge$ or aux). Thus, in Fig. 7B, the concentration of aux is doubled with respect to Fig. 7A, leaving the rest of initial conditions unchanged. We can observe the output is growing faster in this experiment. Now the system would need around 2.7 h (1000 s) to produce 70% of the input concentration. This could be exploited to reduce the signal attenuation when multiple implication rules are cascaded.

3.3. Composition of implications

So far we have seen how the model works with rules that have a single proposition as a premise and a single proposition as a conclusion, like $P \rightarrow Q$. But the model can also be applied recursively, to represent rules like $P \rightarrow (Q \rightarrow R)$ (with a rule as conclusion) or $(P \rightarrow Q) \rightarrow R$ (with a rule as premise).

Fig. 8 shows the implementation of the composite rule $P \rightarrow (Q \rightarrow R)$, which can be rewritten as $\neg P \vee \neg Q \vee R$ (since any implication $A \rightarrow B$ can be rewritten as $\neg A \vee B$). The rewritten expression $\neg P \vee \neg Q \vee R$ is used as the template to design the upper strand of our rule $P \rightarrow (Q \rightarrow R)$. It comprises the following sequence of domains sorted from the 5′ end to the 3′ end: $\neg p_2$, $\neg p_1$, $\neg q_2$, $\neg q_1$, r_1, r_2. Then the bridge strands are designed, so that all the upper strand domains are hybridized except the ones on the ends, which will act as the toeholds of the composite rule $\neg p_2$ and r_2. We divide the single bridge strand with four domains into strands of two domains so that the waste molecules are released and the structure of the output molecules is simpler. Finally, the auxiliary strands are designed as Watson–Crick complements of the bridge strands.

Fig. 8 illustrates the two basic inference mechanisms, modus ponens and modus tollens. When the proposition P is input, it interacts with the composite rule according to modus ponens, restructuring the rule as $Q \rightarrow R$. Symmetrically, if $\neg R$ is the input proposition, the application of modus tollens restructures the original rule as $P \rightarrow \neg Q$ (or $Q \rightarrow \neg P$).

As mentioned above, the rule $P \rightarrow (Q \rightarrow R)$ can be rewritten as $\neg P \vee \neg Q \vee R$. That is also equivalent to $\neg(P \wedge Q) \vee R$, which can also be rephrased as $P \wedge Q \rightarrow R$. Here lies the fundamental interest of the composition operation, as it is useful to representing rules with a conjunction as a premise. Fig. 9 shows how the AND operation is implemented for the rule $P \wedge Q \rightarrow R$ using the composition operation.

Through composition the model is able to represent any type of rule having the format *premises* → *conclusion*, where *conclusion* is made of a single proposition and *premises* is either (1) a conjunction of propositions $(P_1 \wedge \ldots \wedge P_n \rightarrow C)$ or (2) a single proposition $(P \rightarrow C)$. Any other more complex sort of rule can be decomposed into a combination of the previous rules:

- $P \rightarrow C_1 \wedge \ldots \wedge C_n$ can be decomposed into $P \rightarrow C_1$,..., $P \rightarrow C_n$;
- $P_1 \vee \ldots \vee P_n \rightarrow C$ can be decomposed into $P_1 \rightarrow C$,..., $P_n \rightarrow C$;

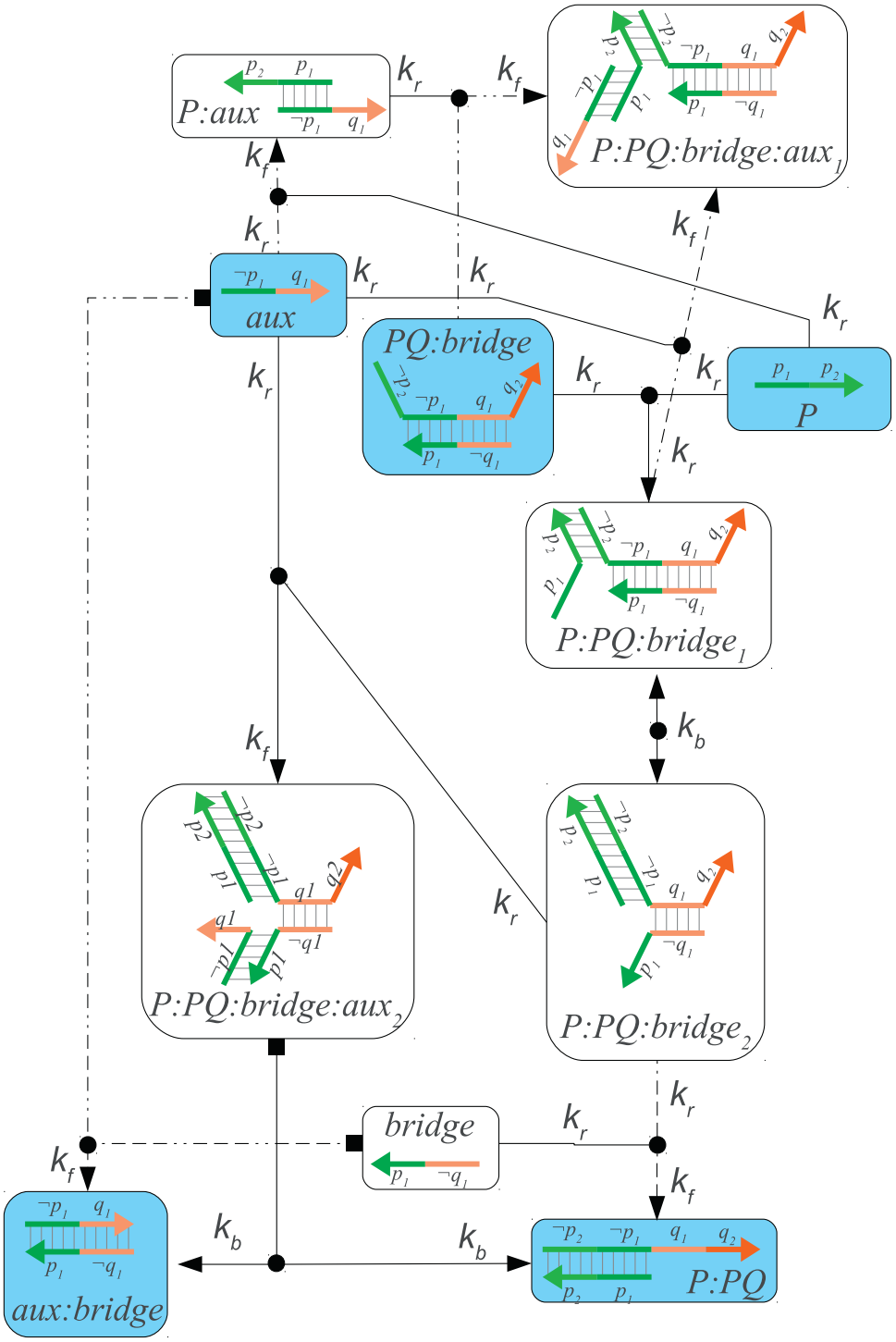

Fig. 6. Reaction kinetics graph – basic inference step. The input molecule (P), the initial conformation of the DNA device (PQ : *bridge* and *aux*) and the output molecules (P : PQ and *aux* : *bridge*) are shadowed to distinguish them from the intermediate molecules of the process. Reactions are depicted with black circles linked to reactants and products with either continuous lines (main reactions) or dashed lines (side reactions). Hybridization–dissociation reaction lines are oriented fixing an arrowhead in the direction of the hybridization (k_f) reaction, while there is no arrowhead in the direction of the dissociation reaction (k_r). Branch migration reaction lines, when bidirectional, assume the same reaction rate (k_b) in either direction, therefore arrowheads are depicted in both sides. Irreversible reaction lines are represented with an arrowhead in the products end and a square in the reactants end.

- $P \rightarrow C_1 \vee \ldots \vee C_n$ can be rewritten as $\neg C \wedge \ldots \wedge \neg C_n \rightarrow \neg P$, which is a rule of type (1).

According to this, we only need two different patterns of a nucleic acid device to implement rules: single premise, illustrated in Fig. 4, and premises connected by a conjunction, illustrated in Fig. 9.

3.3.1. Kinetic model and simulation

The design of the AND operation described above will now be modeled according to mass action kinetic laws. The list of equations below translates the process described in Fig. 9. It contents not only the main reaction flow that drive the steps described there, but also the side reactions representing the crosstalk produced by the

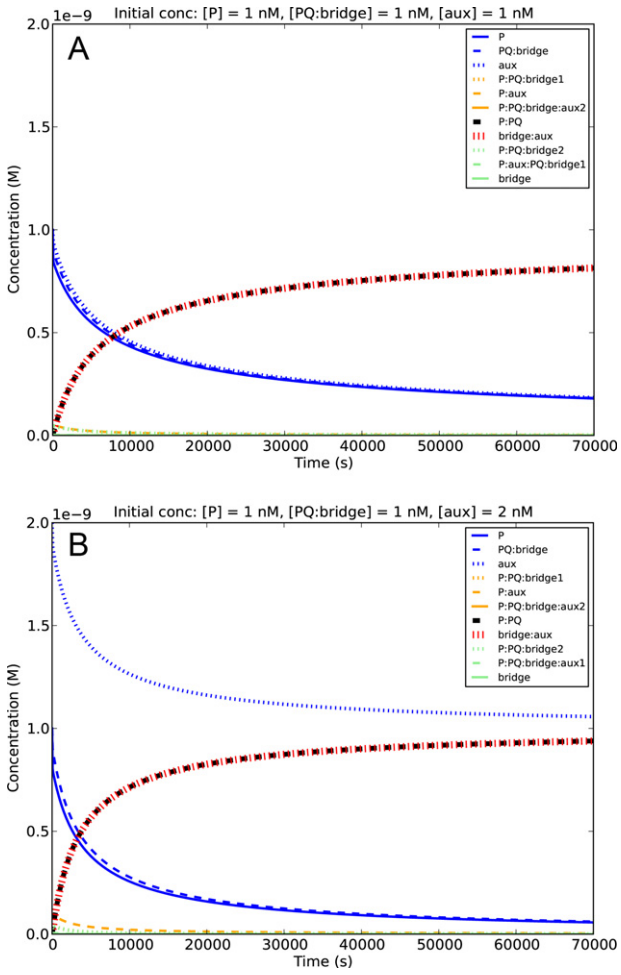

Fig. 7. Simulation results – basic inference step. (A) All the initial concentrations were set to 1 nM. The concentration growth of $P : PQ$ and $bridge : aux$ increases exponentially at the beginning, but becomes flatter as the concentrations of P, $PQ : bridge$ and aux decrease. Concentration of intermediate molecules is always kept to minimum values. The system would need around 8.3 h (30,000 s) to produce 70% of the input concentration. (B) The concentration of aux is doubled with respect to Fig. 7A, leaving the rest of initial conditions unchanged. The output here grows faster. The system would need around 2.7 h (1000 s) to produce 70% of the input concentration.

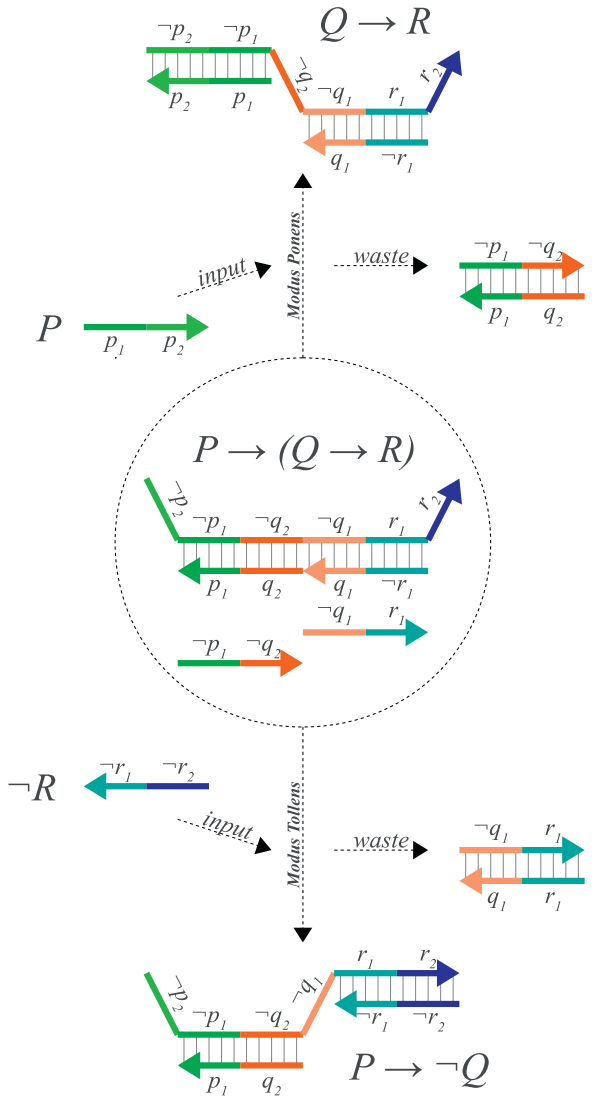

Fig. 8. Rule composition: $P \rightarrow (Q \rightarrow R)$. When the proposition P is input, it interacts with the composite implication according to modus ponens, restructuring it as $Q \rightarrow R$. When the proposition $\neg R$ is input, it interacts with the composite implication according to modus tollens, restructuring it as $P \rightarrow \neg Q$.

rest of potential reactions between the input molecules, the device molecules and the potential intermediate molecules:

Main reaction flow:

$$P + PQR : bridge(PQ) : bridge(QR) \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : PQR : bridge(PQ) : bridge(QR)_1$$

$$P : PQR : bridge(PQ) : bridge(QR)_1 \underset{k_b}{\overset{k_b}{\rightleftharpoons}} P : PQR : bridge(PQ) : bridge(QR)_2$$

$$aux(PQ) + P : PQR : bridge(PQ) : bridge(QR)_2 \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : PQR : bridge(PQ) : bridge(QR) : aux(PQ)_2$$

$$P : PQR : bridge(PQ) : bridge(QR) : aux(PQ)_2 \overset{k_b}{\rightarrow} P : PQR : bridge(QR) + bridge(PQ) : aux(PQ)$$

$$Q + P : PQR : bridge(QR) \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : Q : PQR : bridge(QR)_1$$

$$P : Q : PQR : bridge(QR)_1 \underset{k_b}{\overset{k_b}{\rightleftharpoons}} P : Q : PQR : bridge(QR)_2$$

$$aux(QR) + P : Q : PQR : bridge(QR)_2 \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : Q : PQR : bridge(QR) : aux(QR)_2$$

$$P : Q : PQR : bridge(QR) : aux(QR)_2 \overset{k_b}{\rightarrow} P : Q : PQR + bridge(QR) : aux(QR)$$

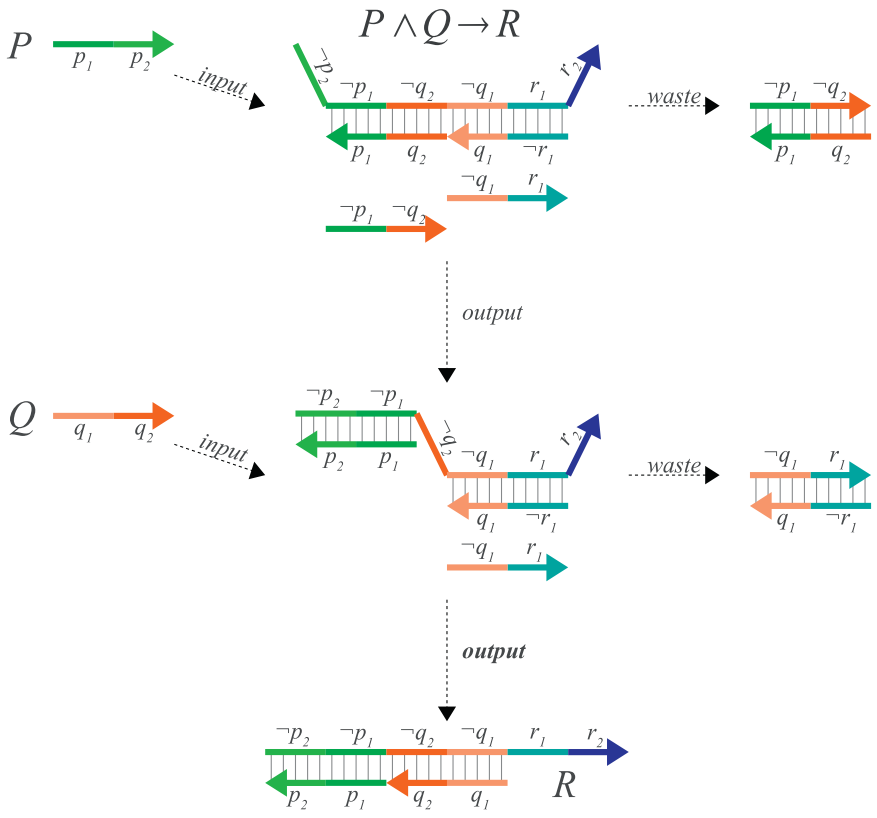

Fig. 9. Rule with a conjunction as a premise: $P \wedge Q \rightarrow R$. The top part of the figure shows the first step of the process, when the input P interacts with $P \wedge Q \rightarrow R$, releasing an intermediate implication equivalent to $Q \rightarrow R$. The process continues in the middle, when input Q interacts with the intermediate implication, releasing the final output R.

Side reactions:

$$P + aux(PQ) \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : aux(PQ)$$

$$P : aux(PQ) + PQR : bridge(PQ) : bridge(QR) \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : PQR : bridge(PQ) : bridge(QR) : aux(PQ)_1$$

$$aux(PQ) + P : PQR : bridge(PQ) : bridge(QR)_1 \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : PQR : bridge(PQ) : bridge(QR) : aux(PQ)_1$$

$$bridge(PQ) + aux(PQ) \overset{k_f}{\rightarrow} bridge(PQ) : aux(PQ)$$

$$Q + aux(QR) \underset{k_r}{\overset{k_f}{\rightleftharpoons}} Q : aux(QR)$$

$$Q : aux(QR) + P : PQR : bridge(QR) \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : Q : PQR : bridge(QR) : aux(QR)_1$$

$$aux(QR) + P : Q : PQR : bridge(QR)_1 \underset{k_r}{\overset{k_f}{\rightleftharpoons}} P : Q : PQR : bridge(QR) : aux(QR)_1$$

$$bridge(QR) + aux(QR) \overset{k_f}{\rightarrow} bridge(QR) : aux(QR)$$

Fig. 10 shows the results of simulating the above equations in Cain with the parameters specified in Section 2.4 and two different initial concentrations of input (P and Q) and implication devices ($PQR : bridge(PQ) : bridge(QR)$, $aux(PQ)$, $aux(QR)$. Only the species achieving quantities greater than 0.1 nM have been plotted. In Fig. 10A, all the initial concentrations were set to 1 nM. The plot for $bridge(PQ) : aux(PQ)$ follows the same growth pattern as the output in Fig. 7A, and $P : PQR : bridge(QR)$ would show the same growth if it would not react with Q and $aux(QR)$. The final output is given by the concentration of $P : Q : PQR$, which is the same as for $bridge(QR) : aux(QR)$. In order to yield a concentration of output equivalent to 70% of the input concentration, the system would need around 19.4 h (70,000 s). As in the example of Fig. 7, this can be improved increasing the concentration of the auxiliary strands in the devices. In Fig. 10B, the concentration of $aux(PQ)$ and $aux(QR)$ is set to 2 nM, leaving the rest of initial conditions unchanged. The output shows a clear improvement in the growth rate of the output, as it only needs around 6.4 h (23,000 s) to produce an amount of $P : Q : PQR$ equivalent to 70% of the input.

3.4. In vitro diagnosis

Benenson et al. (2004) developed a DNA automaton that, after sensing a sequence of mRNA disease indicators, was able to provide a positive or negative diagnosis by either releasing a drug or the respective drug suppressor. This machine was designed as an implementation of a molecular automaton, performing state transitions with the enzyme Fok I. In this article, we propose the implementation of an equivalent system, but using a different computational paradigm as well as different biological processes:

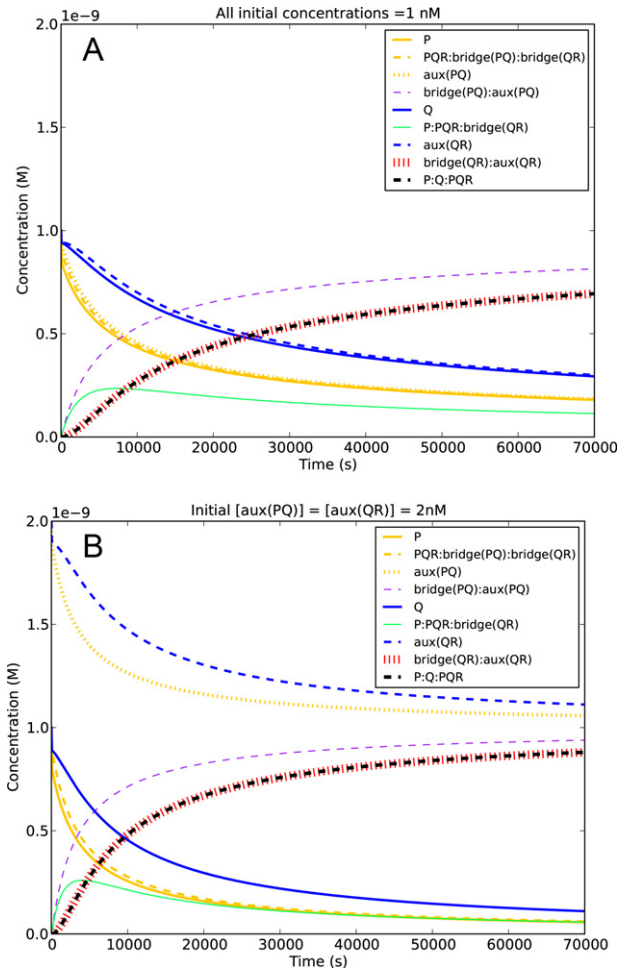

Fig. 10. Simulation results – rule with a conjunction as a premise. Only the species achieving quantities greater than 0.1 nM have been plotted. (A) All the initial concentrations are set to 1 nM. The plot for $bridge(PQ):aux(PQ)$ follows the same growth pattern as the output in Fig. 7A. $P:PQR:bridge(QR)$ would show the same growth if it would not react with Q and $aux(QR)$. The final output is given by the concentration of $P:Q:PQR$, the same as for $bridge(QR):aux(QR)$. The system would need around 19.4 h (70,000 s) to yield an output concentration equivalent to 70% of the input concentration. (B) The concentration of $aux(PQ)$ and $aux(QR)$ is set to 2 nM, leaving the rest of initial conditions unchanged. The output growth rate increases clearly, as it only needs around 6.4 h (23,000 s) to produce an amount of $P:Q:PQR$ equivalent to 70% of the input.

propositional logic instead of a finite state machine, and strand displacement (Seelig et al., 2006; Zhang et al., 2007) instead of enzyme transitions.

As in Benenson et al. (2004), we can use a biological indicator (e.g., high concentration of a specific mRNA) to activate the diagnostic process: the presence of a high level concentration of a specific mRNA strand activates an affirmative input proposition in our system, for instance, $mRNA_1$.

The diagnostic system is composed of a set of implication rules, which encode expert biomedical knowledge using the models described in this article. Depending on the power of the input cue to determine a diagnosis, the implication rules can be designed differently:

- If the presence of a specific input mRNA (or combination of input mRNAs) is sufficient to determine a diagnosis, then the direction of the implication can be set from the input to the diagnosis:

$$mRNA_1(\wedge \ldots \wedge mRNA_n) \rightarrow diagnosis_A$$

- If the absence of a specific input mRNA (or combination of input mRNAs) is sufficient to determine a diagnosis, then the direction of the implication is the same, but the input propositions are negated:

$$\neg mRNA_1(\wedge \ldots \wedge \neg mRNA_n) \rightarrow diagnosis_A$$

- If the presence of a specific input mRNA (or combination of input mRNAs) is not sufficient to determine a diagnosis, then the direction of the implication can only be set from the diagnosis to the input:

$$diagnosis_A \rightarrow mRNA_1(\wedge \ldots \wedge mRNA_n)$$

- If the absence of a specific input mRNA (or combination of input mRNAs) is not sufficient to determine a diagnosis, then the direction of the implication is the same, but the input propositions are negated:

$$diagnosis_A \rightarrow \neg mRNA_1(\wedge \ldots \wedge \neg mRNA_n)$$

The diagnosis takes the form of a special RNA strand, which can either be cascaded to build more complex diagnostic processes, be measured using FRET techniques, or even release a drug as in Benenson et al. (2004).

Now let us look at an example of how the proposed diagnostic machine would work:

- Three different types of mRNA are to be checked in order to release a diagnosis: $mRNA_1$, $mRNA_2$ and $mRNA_3$.
- Three different diagnoses can be emitted: $diagnosis_A$, $diagnosis_B$ and $diagnosis_C$.
- A high concentration of $mRNA_1$ and $mRNA_2$ is sufficient to conclude $diagnosis_A$. This is translated into propositional rules as follows:

$$mRNA_1 \wedge mRNA_2 \rightarrow diagnosis_A$$

- A high concentration of $mRNA_2$ is an indicator of $diagnosis_B$ and $diagnosis_C$, but is not sufficient to emit either diagnosis. This is translated into propositional rules as follows:

$$diagnosis_B \rightarrow mRNA_2$$

$$diagnosis_C \rightarrow mRNA_2$$

- A low concentration of $mRNA_3$ is an indicator of, but is not enough ensure $diagnosis_C$. This is translated into propositional rules as follows:

$$diagnosis_C \rightarrow \neg mRNA_3$$

Now our system senses a high concentration of $mRNA_1$, $mRNA_2$ and $mRNA_3$ as input. Let us analyze how this system would evolve to emit a diagnosis:

- The input propositions $mRNA_1$ and $mRNA_2$ match the premise of the implication $mRNA_1 \wedge mRNA_2 \rightarrow diagnosis_A$; therefore, the modus ponens inference rule is triggered and the output proposition $diagnosis_A$ is released.
- The input proposition $mRNA_3$ is the logical negation of the conclusion in the implication $diagnosis_C \rightarrow \neg mRNA_3$; therefore, the

modus tollens inference rule is triggered releasing the output proposition $\neg diagnosis_C$.

The example has illustrated a very interesting feature of the system: even from a weak implication rule that does not directly determine a diagnosis (like $diagnosis_C \rightarrow \neg\, mRNA_3$), we can take advantage of modus tollens to infer interesting knowledge (like the exclusion of diagnosis C, $\neg diagnosis_C$).

Both output propositions can be designed to activate a specific sort of fluorescence so that the system's output can be recognized and measured.

3.5. Error measurement

The previous sections described the expected behavior of our devices: if our system is made of the rule $P \rightarrow Q$, and we give P as the input, the expected output is Q. However, there is also a probability of an input other than P (i.e. R) outputting Q. In this section, a relative error probability will be defined, as a ratio between the reaction rate that releases Q when the input is P and the reaction rate that releases Q when the input is R (other than P).

The reaction rates for each case can be easily estimated using the *bimolecular reaction model* (Zhang et al., 2007) described in Fig. 1. Based on the parameters m (invading toehold length) and n (incumbent toehold length), we can refer to Zhang et al. (2007) to get a rough estimate of $k_{\{m,n\}}$.

The ideal domain length that optimizes the length of the toeholds should be around 6 (Zhang and Winfree, 2009):

- Longer domains do not significantly increase their hybridization rate to a complementary domain, whereas their dissociation rate decreases (an undesired behavior for a toehold).
- With shorter domains, the duplexes resulting from complementary strands, each composed of two consecutive domains, would not be stable enough to be considered an irreversible reaction.
- Length 6 offers the best compromise of a good hybridization rate without decreasing the dissociation rate, which is good behavior for a toehold. Besides, the duplexes resulting from complementary strands, each composed of two consecutive domains, are stable enough to be considered an irreversible reaction.

Therefore, we will consider toeholds of length 6 in the calculation of the relative error probabilities.

The following functions will help us to formally assess the probability:

- $F(input,\ rule) = output$: this function denotes the output of a rule when a specific input is given. For instance, $F(P,\ P \rightarrow Q) = Q$ describes the expected behavior of rule $P \rightarrow Q$ when P is given as input.
- $Pr(event) = probability$: this function denotes probabilities. For example, $Pr(F(P,\ P \rightarrow Q) = Q) = 1$ would indicate that the probability of Q being output when the input P operates with the rule $P \rightarrow Q$ is 1.
- $Hyb(strand_1, \ldots, strand_n)$: this function denotes an hybridization event between the strands 1 and n. For instance, $hyb(bridge, aux)$ would denote a hybridization reaction between the bridge and auxiliary strands

Let us assume we have a proposition P almost equal to R, meaning that $p_1 = r_1$ and r_2 only differs from p_2 by one base, situated at its 3′ end (marked in yellow in Fig. 11); this assumption represents the worst case scenario, as there is no other proposition that is more similar to P. We will assume that the length of each domain (p_i, q_i, r_i) is 6. According to the above bimolecular reaction model and ignoring the effect of the auxiliary strand on the bridge strand, R would release Q from the device $P \rightarrow Q$ with a reaction rate $K_{\{6,5\}} = 5 \times 10^4\ M^{-1}\ s^{-1}$; the corresponding rate for P would be $K_{\{6,6\}} = 5 \times 10^5\ M^{-1}\ s^{-1}$; therefore, we can calculate $Pr(F(R,\ P \rightarrow Q) = Q,\ R \neq P) = K_{\{6,5\}}/K_{\{6,6\}} = 1/10$. The plot in Fig. 11 shows the variation of the error probability for our proposition R, when $p_1 = r_1$ and r_2 comprises an invading toehold of length n.

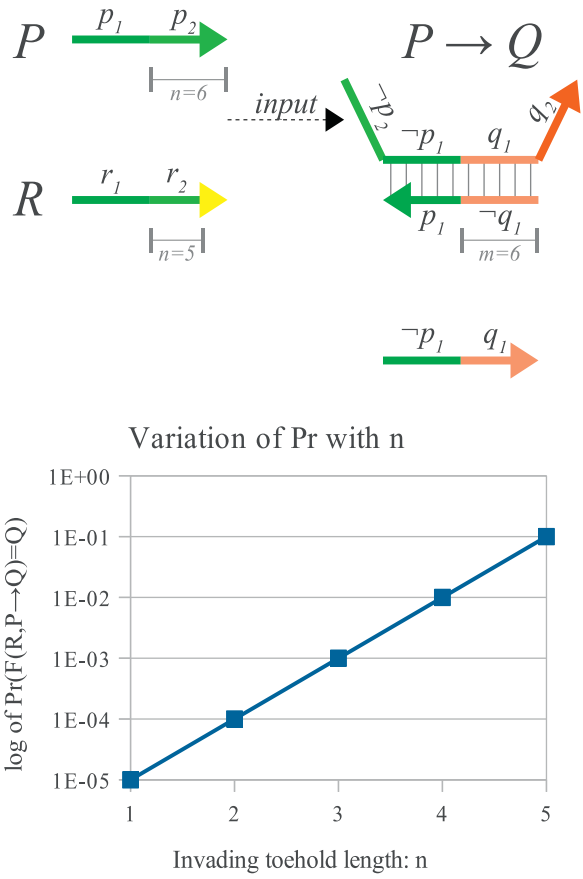

Fig. 11. Variation of $Pr(F(R, P \rightarrow Q))$ with $p_1 = r_1$ and $p_2 \neq r_2$. Proposition R is almost identical to P; its only difference is one nucleotide located at the 3′ end of R, marked in yellow in the diagram. Therefore, when modeling the potential toehold exchange reaction between R and $P \rightarrow Q$, we have to consider an invading toehold of length $n = 5$ and an incumbent toehold of length $m = 6$, meaning a reaction rate $K_{\{6,5\}} = 5 \times 10^4\ M^{-1}\ s^{-1}$; the corresponding rate for P would be $K_{\{6,6\}} = 5 \times 10^5\ M^{-1}\ s^{-1}$; therefore, we can calculate the relative probability of error as $Pr(F(R,\ P \rightarrow Q) = Q,\ R \neq P) = K_{\{6,5\}}/K_{\{6,6\}} = 1/10$. The plot at the bottom shows the variation of the probability of error for our proposition R, when $p_1 = r_1$ and r_2 contains an invading toehold of length n. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

If we take into account the effect of the auxiliary strand, we must build the probability function differently. Instead of one reaction modeled as toehold exchange, two toehold mediated strand displacement reactions have to be considered. Therefore it should be stated as a product of the following factors:

- Probability of R hybridizing with $\neg p_2$ and $\neg p_1$, and displacing the domain p_1 in the bridge strand. It can be estimated as the following quotient: $Pr(hyb(R, P \rightarrow Q)) = K_{\{0,5\}}/K_{\{0,6\}} = 1/10$.
- Probability of the auxiliary strand binding the domain p_1 in the bridge strand and completing a toehold mediated strand displacement process. It can be estimated as the following quotient: $Pr(hyb(bridge, aux)) = K_{\{0,6\}}/K_{\{0,6\}} = 1$.

The result is also 1/10, so both methods are equivalent. The plot resulting of this method would be as shown in Fig. 11.

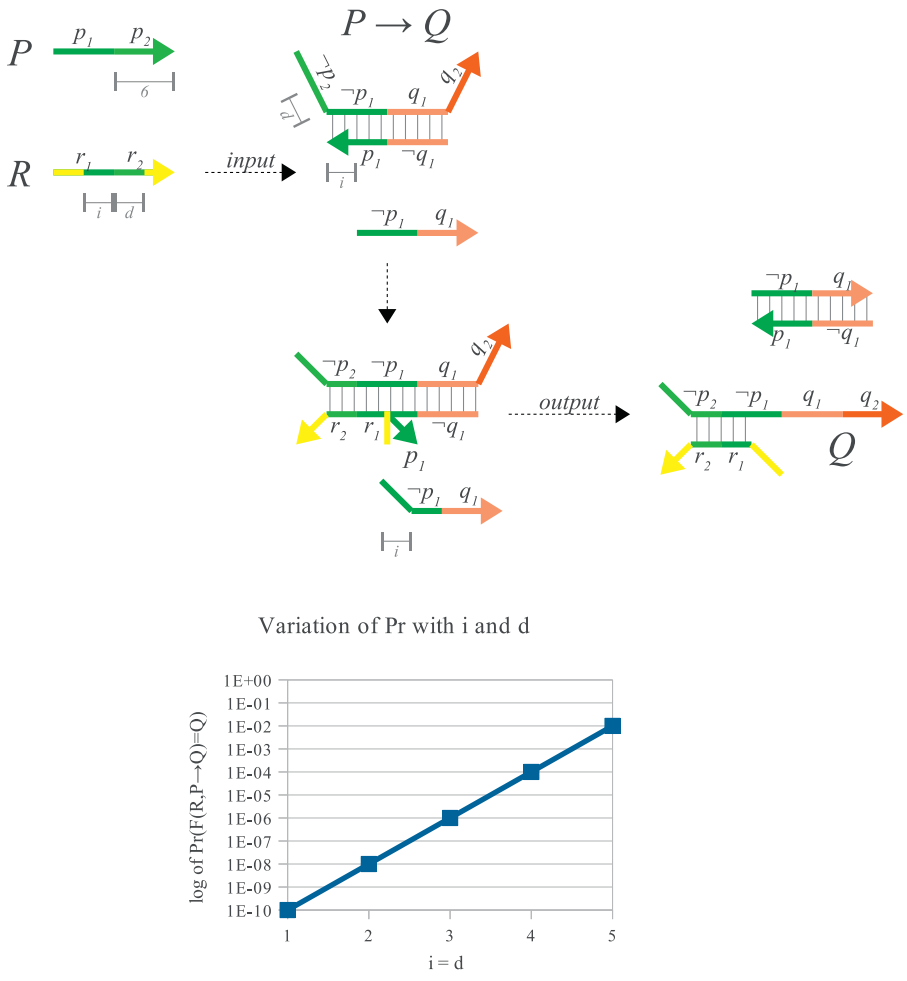

Fig. 12. Variation of $Pr(F(R, P \rightarrow Q)$ with $p_1 \neq r_1$ and $p_2 \neq r_2$. This analyzes the best case scenario, assuming that R has the most unfavorable configuration to supplant P, meaning that r_1 and p_1 have only the leftmost base in common, and r_2 and p_2 have only the rightmost base in common. The bases of R that differ from P are colored in yellow, whereas the common bases are confined to the center of the strand and have length $i = 1$ and $d = 1$. The rest of the diagram illustrates how such a proposition R would release the output Q from an implication $P \rightarrow Q$. The plot at the bottom of the figure shows how the probability of error increases when i and d increase. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

Let us now analyze the best case scenario, assuming that R has the most unfavorable configuration to supplant P, meaning that r_1 and p_1 only have the leftmost base in common, and r_2 and p_2 only have the rightmost base in common. In Fig. 12 the bases of R that differ from P are colored yellow, whereas the common bases are at the center of the strand and have length $i = 1$ and $d = 1$. The rest of the diagram illustrates how such a proposition R would release the output Q from an implication $P \rightarrow Q$, and its probability is defined as follows:

$$Pr(F(R, P \rightarrow Q) = Q, R \neq P) = Pr(hyb(R, P \rightarrow Q)) \times Pr(hyb(bridge, aux)) = \frac{K_{\{0,d\}}}{K_{\{0,6\}}} \times \frac{K_{\{0,i\}}}{K_{\{0,6\}}} = \frac{K_{\{0,1\}}}{K_{\{0,6\}}} \times \frac{K_{\{0,1\}}}{K_{\{0,6\}}} = 10^{-10}.$$

The plot at the bottom of Fig. 12 shows how the error probability increases when i and d increase.

Finally, we can calculate the average probability of a generic proposition R, other than P, releasing the output Q of a rule $P \rightarrow Q$. This can be done using a weighted sum of the probabilities of each possible configuration of R (combining values of d ranging from 1 to 5, and values of i ranging from 1 to 6); the weight will be the probability of such an occurrence happening. This can be denoted as follows: $Pr(\exists R) = 1/4^{i+d}$.

The average probability follows:

$$\begin{aligned} Pr_{avg} &= \sum_{i=1,d=1}^{5,6} Pr(F(R, P \rightarrow Q) = Q, R \neq P) \\ &= \sum_{i=1,d=1}^{5,6} Pr(\exists R) \times Pr(hyb(R, P \rightarrow Q)) \times Pr(hyb(bridge, aux)) \\ &= \sum_{i=1,d=1}^{5,6} \frac{1}{4^{i+d}} \times \frac{K_{\{0,l\}}}{K_{\{0,6\}}} \times \frac{K_{\{0,r\}}}{K_{\{0,6\}}} \simeq 6,53 \times 10^{-8}. \end{aligned}$$

We can see that the value is much closer to the best case scenario (10^{-10}) than the worst case scenario (1/10).

4. Discussion

The DNA biosensor presented here operates as an inference machine at least as expressive as the system implemented in Ran et al. (2009). Although our model cannot compete in speed with Shapiro et al.'s system, it has two noteworthy features: (1) absence

of restriction enzymes and (2) explicit representation of the logical negation.

The ability to explicitly represent negations increases the power of the rules. The very same implication $P \rightarrow Q$ can be triggered by the presence of the premise (applying modus ponens) and by the presence of the negated conclusion (applying modus tollens). It also means that our method can offer a negative proposition as output. This could be an improvement on Ran et al. (2009), since it implicitly acts as an error correction mechanism. This dual encoding can help us execute more robust computations and longer cascades of inference. Let us imagine that we are expecting output Q, but the dissolution contains strands with $\neg Q$ spontaneously released by error from other rules or the environment; the error strands would hybridize with the solution strands, and since the solution strands would have higher concentration, the right signal would be propagated free of noise. This error cancellation feature, together with the iteration and recursion properties, increases the scalability of the model.

Leak reactions, that end up hybridizing bridge and auxiliary strands in absence of any input, have not been included in our kinetic model. Even though leak rate constants can be up to a million times slower than their corresponding desired reaction (Zhang et al., 2007; Zhang and Winfree, 2009), this type of reaction will need to be taken into account in future implementations *in vitro* of the model, since it will be key to determine the lifetime of the gates.

Another potential drawback of leak reactions has to do with the stability of transitive rules, such as $P \rightarrow Q$ and $Q \rightarrow R$, in the same dissolution: it could happen that, while the toeholds q_2 and $\neg q_2$ were reversibly bond (see Fig. 5), a leak between the bridge and auxiliary strands of $Q \rightarrow R$ could trigger the release of R before Q is released from $P \rightarrow Q$. A potential way to keep the leak rate down would be to shorten the auxiliary strands by one or two nucleotides on each end, keeping same length for bridge strands.

The analysis of the probability of error denotes a high robustness. Working with a fully synthetic system, we can fix an upper error bound at model strand design time. Let us imagine that we design all the propositions ensuring each 6-base domain has a Hamming distance of 1 from the rest; in the worst case scenario, where a proposition R with 2 domains differs from the expected premise by only one base at the 3′ end and another one at the 5′ end, the probability of error would be 10^{-2}. If the domains are guaranteed to have a Hamming distance of 2, the probability of error would be 10^{-4}, etc. Hence the upper bound of the probability of error could be fixed at design time.

5. Conclusions

This paper proposes a simple biosensor model made of nucleic acids. The model is able to perform logical deductions applying inference rules (modus ponens and modus tollens). No restriction enzymes are used. Besides DNA, our encodings can also take any type of RNA molecule as a substrate (as shown in Section 3.4). We think it can leverage interesting applications in the area of synthetic RNA circuits (Davidson and Ellington, 2007; Rinaudo et al., 2007; Xie et al., 2010).

Previous models (Ran et al., 2009; Reif and Sahu, 2009) represent the logical true value with a strand of nucleic acid and the false value with an absence of that strand. Instead, our models encode the respective false value with the Watson–Crick complement of the DNA strand associated with the true value. The interest of this encoding is twofold: (1) it introduces an implicit error cancellation mechanism, which increases the system scalability enabling longer inference cascades with a bounded and controllable signal-noise relation, and (2) it allows the same rule to be used in forward inference (modus ponens) or backward inference (modus tollens), providing the option of validly outputting negated propositions (e.g. "diagnosis A excluded"). The output signal of the devices represents a diagnosis, which can either be measured using FRET techniques, be cascaded as input for another logical deduction with different rules, or even be a drug which is administered in response to a set of symptoms.

Acknowledgements

This research was partially supported by the European Commission funded 7th Framework Programme (FET Proactive area) BACTOCOM project, Spanish Ministry of Science and Innovation project TIN2009-14421, and Madrid's Regional Government. We would also like to thank Rachel Elliott for polishing the English, and Jesús Miró-Bueno for helping with the simulations tools.

References

Adleman, L., 1994. Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024.
Benenson, Y., Adar, R., Paz-Elizur, T., Livneh, Z., Shapiro, E., 2003. DNA molecule provides a computing machine with both data and fuel. Proceedings of the National Academy of Sciences of the United States of America 100, 2191–2196.
Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E., 2004. An autonomous molecular computer for logical control of gene expression. Nature 429, 423–429, doi:10.1038/nature02551.
Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E., 2001. Programmable and autonomous computing machine made of biomolecules. Nature 414, 430–434, doi:10.1038/35106533.
Cardelli, L., 2009. Strand algebras for dna computing. In: Deaton, R., Suyama, A. (Eds.), In: DNA Computing and Molecular Programming, vol. 5877. Springer, Berlin, Heidelberg, pp. 12–24 (Chapter 2).
Cockroft, B., Ghadiri, M., 2007. Modular multi-level circuits from immobilized DNA-Based logic gates. Journal of the American Chemical Society 129, 14875–14879.
Davidson, E., Ellington, A., 2007. Synthetic RNA circuits. Nature and Chemical Biology 3, 23–28, doi:10.1038/nchembio846.
Hagiya, M., Arita, M., Kiga, D., Sakamoto, K., Yokoyama, S., 1997. Towards parallel evaluation and learning of boolean μ-formulas with molecules. In: Proceedings of Third Annual Meeting on DNA Based Computers 48, pp. 105–114.
Kobayashi, S., 1999. Horn clause computation with DNA molecules. Journal of Combinatorial Optimization 3, 277–299.
Lipton, R., 1995. DNA solution of hard computational problems. Science 268, 542–545.
Lloyd, J.W., 1987. Foundations of Logic Programming, 2nd extended ed. Springer-Verlag New York, Inc., New York, NY, USA.
Macdonald, J., Stefanovic, D., Stojanovic, M., 2008. DNA computers for work and play. Scientific American 299, 84–91.
Qian, L., Winfree, E., 2009. A simple DNA gate motif for synthesizing large-scale circuits. In: Goel, A., Simmel, F., Sosík, P. (Eds.), DNA Computing. Lecture Notes in Computer Science, vol. 5347. Springer, Berlin, Heidelberg, pp. 70–89.
Qian, L., Winfree, E., 2011a. Scaling up digital circuit computation with DNA strand displacement cascades. Science 332, 1196–1201.
Qian, L., Winfree, E., 2011b. A simple DNA gate motif for synthesizing large-scale circuits. Journal of The Royal Society Interface.
Qian, L., Winfree, E., Bruck, J., 2011. Neural network computation with DNA strand displacement cascades. Nature 475, 368–372.
Ran, T., Kaplan, S., Shapiro, E., 2009. Molecular implementation of simple logic programs. Nature Nanotechnology 4, 642–648.
Reif, J.H., Sahu, S., 2009. Autonomous programmable DNA nanorobotic devices using DNAzymes. Theoretical Computer Science 410, 1428–1439.
Rinaudo, K., Bleris, L., Maddamsetti, R., Subramanian, S., Weiss, R., Benenson, Y., 2007. A universal RNAi-based logic evaluator that operates in mammalian cells. Nature Biotechnology 25, 795–801, doi:10.1038/nbt1307.
Rodríguez-Patón, A., Larrea, J.M., Sainz de Murieta, I., 2010. Inference with DNA molecules. In: Calude, C.S., Hagiya, M., Morita, K., Rozenberg, G., Timmis, J. (Eds.), In: Unconventional Computation, vol. 6079. Springer, Berlin, Heidelberg, p. 192 (Chapter 25).
Rodríguez-Patón, A., Sainz de Murieta, I., Sosík, P., 2011. Autonomous resolution based on DNA strand displacement. In: Cardelli, L., Shih, W. (Eds.), DNA Computing and Molecular Programming. Lecture Notes in Computer Science, vol. 6937. Springer, Berlin, Heidelberg, pp. 190–203.
Rose, J.A., Komiya, K., Yaegashi, S., Hagiya, M., 2006. Displacement Whiplash PCR: optimized architecture and experimental validation. In: Mao, C., Yokomori, T. (Eds.), DNA Computing. Springer, pp. 393–403.
Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T., Hagiya, M., 2000. Molecular computation by DNA hairpin formation. Science 288, 1223–1226.
Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E., 2006. Enzyme-free nucleic acid logic circuits. Science 314, 1585–1588.

Smolke, C.D., 2009. It's the DNA that counts. Science 324, 1156–1157.
Soloveichik, D., Seelig, G., Winfree, E., 2008. DNA as a universal substrate for chemical kinetics. In: Goel, A., Simmel, F.C., Sosík, P. (Eds.), DNA. Springer, pp. 57–69.
Sterling, L., Shapiro, E., 1994. The Art of PROLOG: Advanced Programming Techniques. The MIT Press.
Takahashi, K., Yaegashi, S., Kameda, A., Hagiya, M., 2006. Chain reaction systems based on loop dissociation of DNA. DNA Computing 3892, 347–358.
Wasiewicz, P., Janczak, T., Mulawka, J.J., Plucienniczak, A., 2000. The inference based on molecular computing. Cybernetics and Systems 31, 283–315.
Wasiewicz, P., Malinowski, A., Nowak, R., Mulawka, J.J., Borsuk, P., Weglenski, P., Plucienniczak, A., 2001. DNA computing: implementation of data flow logical operations. Future Generation Computer Systems 17, 361–378.
Xie, Z., Liu, S.J., Bleris, L., Benenson, Y., 2010. Logic integration of mRNA signals by an RNAi-based molecular computer. Nucleic Acids Research 38, 2692–2701.
Zhang, D.Y., Turberfield, A.J., Yurke, B., Winfree, E., 2007. Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318, 1121–1125.
Zhang, D.Y., Winfree, E., 2009. Control of DNA strand displacement kinetics using toehold exchange. Journal of the American Chemical Society 131, 17303–17314.