Sequential Decision-Making in DNA: From POMDPs to Molecular Controllers

Antonio Llano
Computer Science
Stanford University
llano@stanford.edu

Shobhit Agarwal
Computer Science
Stanford University
shobhit.agarwal@stanford.edu

Ethan Goodhart
Computer Science
Stanford University
goodhart@stanford.edu

***Abstract*—Chronic inflammatory diseases, such as rheumatoid arthritis, are managed today by repeated dose adjustments based on noisy biomarkers and clinician intuition. This is inherently a sequential decision-making problem under uncertainty: clinicians trade off flare risk against drug toxicity over months to years. At the same time, there is a growing body of work on "computing drugs," including DNA-based logic gates and cell therapies that implement Boolean AND/OR/NOT decisions over molecular inputs. These systems prove that biochemical substrates can perform computation in vivo, but their behavior is typically static: a fixed logic function of current signals, not an adaptive policy that evolves with the disease.**

We propose an end-to-end pipeline that turns a POMDP treatment strategy into a DNA circuit that could, in principle, run directly in the biochemical environment it controls. We first model disease and toxicity as a partially observable Markov decision process, with biomarker categories as observations and discretized drug doses as actions. Dynamic programming on the fully observed model yields an optimal state–action value function, which defines a belief-space dosing policy. We then compress this policy into a finite-state controller by clustering beliefs and learning a Mealy machine whose nodes encode actions and whose transitions are driven by observed biomarker symbols. This FSC is compiled into a DNA strand-displacement (DSD) circuit, where state strands, input strands, and strand-displacement gates collectively implement the same discrete-time policy as a chemical reaction network. Finally, we design a topology-aware reinforcement-learning (RL) environment for nucleotide sequence optimization, using NUPACK and Peppercorn as thermodynamic and kinetic oracles [6], [7]. We validate this RL "sequence gym" on a hairpin benchmark and outline its application to FSC-derived circuits. We demonstrate the full pipeline on a stylized inflammatory-disease simulator, automatically generate Peppercorn-compatible DSD designs, and discuss how RL-guided sequence design can yield low-leak, robust *in vivo* treatment controllers.

***Index Terms*—DNA strand-displacement, finite-state controllers, POMDPs, reinforcement learning, DNA computing, molecular programming**

I. Introduction

DNA strand-displacement (DSD) circuits have demonstrated Boolean computation (AND/OR/NOT gates), signal restoration, and even small neural networks implemented purely in chemistry [4], [5]. In parallel, logic-gated cell therapies and drug carriers activate treatment only under specific combinations of biomarkers [8]. These "computing drugs" show that biochemical substrates can evaluate predefined logic functions over molecular inputs *in situ*. However, their behavior is typically *static*: a fixed combinational map from current inputs to outputs.

Real-world biosensing and therapy, especially in chronic inflammatory disease, demands *sequential* decision-making. A therapeutic implant monitoring inflammatory biomarkers over days must accumulate evidence before committing to drug release, balancing false-positive risk against treatment delay. Clinicians essentially solve a partially observable, multi-period decision problem at each visit: they see noisy biomarkers and symptoms, maintain an implicit belief over disease and toxicity states, and choose a dose adjustment. Yet current molecular implementations are largely one-shot or memoryless.

A. From belief updates to finite-state controllers

A natural starting point is to model disease management as a partially observable Markov decision process (POMDP), with latent states capturing disease activity and toxicity, actions as drug doses, and observations as noisy biomarkers. In principle, one could attempt to implement Bayesian belief updates directly in DNA, maintaining probability distributions $b_t(s)$ and updating them with each observation. Prior work has demonstrated one-shot Bayesian inference in DNA for simple networks, but extending this to multi-episode belief tracking faces fundamental biophysical barriers:

- **DNA degradation and turnover** makes it difficult to store a precise probabilistic state over long time scales.
- **Drift and leak** accumulate over sequential strand-displacement reactions, degrading computational fidelity.

Rather than computing beliefs *in DNA*, we adopt a compilation perspective: compute an optimal POMDP policy *in silico*, then compress it into a finite-state controller (FSC) with a small number of discrete memory states. FSCs map directly to DSD cascades: each controller node corresponds to a molecular state, and observations trigger strand-displacement reactions that transit between nodes and emit dosing decisions.

B. Pipeline overview

Fig. 1 summarizes our end-to-end pipeline. At a high level:

1) **POMDP layer.** We formulate a stylized chronic inflammatory disease as a POMDP. Dynamic programming on the fully observed MDP yields an optimal state–action

Identify applicable funding agency here. If none, delete this.

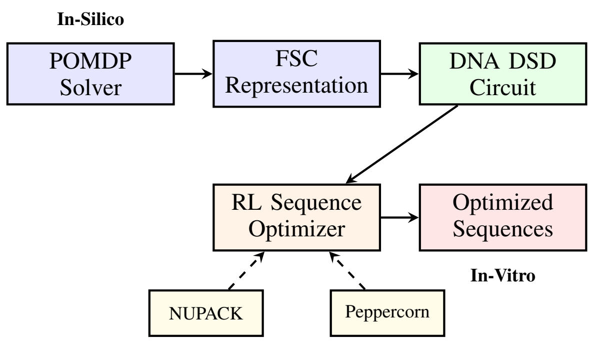


Fig. 1: End-to-end pipeline from POMDP to molecular controller and topology-aware RL sequence optimization.

value function, which we lift to belief space to obtain a dosing policy $\pi(b)$.

2) **FSC extraction.** We roll out $\pi(b)$ to collect belief trajectories and cluster beliefs with k-means, yielding a finite-state controller (Mealy machine) whose nodes encode actions and whose observation-labeled transitions approximate the belief-based policy.
3) **DSD compilation.** We map the FSC to a DSD architecture: state strands represent controller nodes, observation strands encode biomarker symbols, and gate complexes correspond to controller transitions. The output is a domain-level DSD description in Peppercorn's `.pil` language.
4) **RL sequence optimization.** Given a DSD topology, we define a topology-aware RL "gym" whose state includes nucleotide sequences and structural/kinetic features from NUPACK and Peppercorn, actions are point mutations, and rewards favor correct structures, strong intended reactions, and low leak. An actor–critic policy is trained via PPO to navigate sequence space.

Our contributions are: (i) a POMDP model and approximate belief-based policy for a chronic inflammatory disease, (ii) a clustering-based extraction of small FSCs that preserve most of the policy's performance, (iii) a compilation from FSCs to domain-level DSD circuits, and (iv) a topology-aware RL environment for sequence design, validated on a hairpin benchmark and designed to extend to FSC-derived circuits.

II. Related Work

A. POMDPs in medical decision-making

POMDPs formalize sequential decision-making under partial observability [1]. They have been applied to various medical problems, including chronic disease management and intensive care decision support [2], [3]. These works assume digital controllers (software agents) that emit recommendations, with no attempt to embed the policy in the biochemical substrate itself. We adopt the POMDP machinery but aim at a molecular implementation.

B. DNA computing and strand-displacement circuits

DSD has been used to implement logic circuits, oscillators, and neural networks in vitro [4], [5]. NUPACK [6] supports thermodynamic analysis and sequence design; Peppercorn [7] enumerates reaction networks and estimates reaction rates. Prior work has also demonstrated one-shot Bayesian inference and simple state machines in DNA. However, most circuits implement static logic or finite-depth computations; they do not encode long-lived policies interacting with a stochastic environment.

C. Molecular logic for therapeutics

Logic-gated CAR-T cells and drug delivery systems deploy treatment only when specific antigen combinations are detected [8], [9]. These systems demonstrate context-dependent therapy but are typically memoryless. Our finite-state controller maintains discrete internal memory (nodes) that capture aspects of disease and toxicity history and condition future dosing decisions.

D. Finite-state controllers for POMDPs

Finite-state controllers (policy graphs) are a classical representation for approximate POMDP solutions [2], [10]. Nodes encode controller memory; edges are labeled by observations. Previous work learns FSCs via policy iteration or search, primarily for digital agents. We obtain FSCs by clustering belief trajectories induced by an approximate optimal policy and target a physical DSD realization.

E. RL and sequence design

Reinforcement learning has been applied to RNA and protein design, where agents propose sequences and receive rewards from folding or binding oracles [11]. In nucleic-acid design, NUPACK has been combined with heuristic optimization for DSD circuits [6]. We extend this line of work by embedding sequence design in an RL environment that is explicitly *topology-aware*: the RL agent sees features derived from the specific FSC-derived DSD architecture and receives feedback from both NUPACK and Peppercorn.

III. Disease Model and Belief Features

A. State, action, and observation spaces

We define a small POMDP intended to capture the qualitative tradeoffs of chronic inflammatory disease management. The latent state $s = (D, T)$ consists of:

- disease activity $D \in \{0, 1, 2\}$ (remission, moderate disease, severe flare);
- toxicity status $T \in \{0, 1\}$ (acceptable vs. high toxicity).

The action space is $\mathcal{A} = \{0, 1, 2\}$: no drug, low dose, high dose. The observation space $\mathcal{O} = \{0, 1, 2, 3\}$ is defined by discretizing two noisy binary flags: high vs. low inflammation and acceptable vs. bad toxicity, yielding four biomarker categories.

B. *Transition, observation, and reward models*

The transition model $T(s, a, s')$ factorizes into:

- $P(D'|D, a)$ capturing flare risk and treatment efficacy: under no drug, disease tends to worsen with some spontaneous remission; under low and high dose, disease tends to improve, with stronger effects at high dose.
- $P(T'|T, a)$ capturing toxicity: high-dose therapy increases toxicity, while toxicity can recover when off drug.

The observation model $O(s, o)$ is constructed by mapping the latent flags "disease high?" and "toxicity bad?" to four symbols via independent noise parameters, representing imperfect biomarkers.

The reward function penalizes disease, toxicity, and drug use:

$$R(s, a) = -(\alpha D + \beta T + \lambda a), \tag{1}$$

with $\beta > \alpha > 0$ and $\lambda > 0$ so that toxicity is weighted more heavily than disease, and higher doses incur higher costs.

C. *Belief state as feature vector*

The information state for optimal control is the belief $b_t(s) = P(s_t = s \mid o_{0:t}, a_{0:t-1})$, which evolves under Bayes' rule:

$$b_{t+1}(s') \propto O(s', o_t) \sum_s b_t(s)\, T(s, a_t, s'). \tag{2}$$

Since $|\mathcal{S}| = 6$, each belief is a 6-dimensional probability vector. We treat b_t itself as the feature representation for downstream clustering and FSC extraction.

D. *Model parameterization*

For concreteness, we instantiate the transition, observation, and reward parameters to capture a stylized but interpretable inflammatory-disease domain. Disease dynamics encode (i) an increased flare risk in the absence of treatment, with some chance of spontaneous improvement, and (ii) progressively stronger therapeutic effects as dose increases. Toxicity dynamics encode an increased risk of high-toxicity states under treatment and partial recovery when treatment is withheld. Observations are generated from imperfect biomarkers for inflammation and toxicity via independent noise channels. Formally, we use:

- flare and remission parameters $p_{\text{flare,no drug}} = 0.3$, $p_{\text{spont.remission}} = 0.2$;
- low- and high-dose improvement probabilities $p_{\text{improve,low}} = 0.6$, $p_{\text{improve,high}} = 0.8$;
- toxicity induction and recovery probabilities $p_{\text{tox,low}} = 0.15$, $p_{\text{tox,high}} = 0.35$, $p_{\text{tox,recover}} = 0.3$;
- inflammation and toxicity observation noise $p_{\text{obs noise,inflam}} = 0.2$, $p_{\text{obs noise,tox}} = 0.1$.

The reward weights are set to $(\alpha, \beta, \lambda) = (2.0, 4.0, 0.3)$ so that toxicity is penalized twice as heavily as disease and high-dose therapy incurs a modest cost. We use a discount factor $\gamma = 0.95$ for dynamic programming on the fully observed MDP. These choices yield qualitatively reasonable episodes in which the controller trades off flare risk against cumulative toxicity.

IV. Methods

A. *Approximate POMDP solution via MDP value iteration*

We approximate the POMDP solution by solving the fully observed MDP and lifting the result to belief space. Value iteration iterates

$$V_{k+1}(s) = \max_{a \in \mathcal{A}} \left[R(s, a) + \gamma \sum_{s'} T(s, a, s') V_k(s') \right] \tag{3}$$

until convergence to V^*. The optimal state–action value function is

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s'} T(s, a, s') V^*(s'), \tag{4}$$

and the MDP policy is $\pi_{\text{MDP}}(s) = \arg\max_a Q^*(s, a)$.

Given Q^*, we define a belief-based policy by expectation:

$$Q^*(b, a) = \sum_s b(s)\, Q^*(s, a), \tag{5}$$

$$\pi(b) = \arg\max_a Q^*(b, a). \tag{6}$$

This ignores the value of information but produces a simple and interpretable dosing policy over beliefs. In our implementation, value iteration is run with $\gamma = 0.95$ and a convergence tolerance of 10^{-6} starting from $V_0 \equiv 0$.

B. *Belief trajectory collection*

To prepare for FSC extraction, we simulate many episodes under $\pi(b)$:

1) Initialize b_0 uniformly over $\mathcal{S}$ and draw s_0 accordingly.
2) At each time t, set $a_t = \pi(b_t)$.
3) Sample $s_{t+1} \sim T(s_t, a_t, \cdot)$ and $o_t \sim O(s_{t+1}, \cdot)$.
4) Update b_{t+1} via the belief update equation.

We record (b_t, a_t, o_t, b_{t+1}) for all time steps. This dataset captures how the approximate optimal policy uses information over time. In the rheumatoid-arthritis case study below, we use 500 episodes of length $H = 50$ steps each, with independent initial states and observation noise draws across episodes.

C. *Finite-state controller extraction*

A finite-state controller (FSC) is a triple $(\mathcal{N}, a(\cdot), \tau(\cdot, \cdot))$, where $\mathcal{N} = \{0, \ldots, K-1\}$ is a finite set of nodes, $a : \mathcal{N} \to \mathcal{A}$ assigns an action to each node, and $\tau : \mathcal{N} \times \mathcal{O} \to \mathcal{N}$ is a deterministic transition function. The FSC maintains a node n_t, emits $a(n_t)$ as the action, observes o_t, and updates $n_{t+1} = \tau(n_t, o_t)$.

We construct an FSC from belief trajectories as follows:

1) **Clustering beliefs:** Apply k-means clustering to the set $\{b_t\}$, yielding K clusters with centroids c_k; each cluster index k defines a controller node.
2) **Action assignment:** For each cluster k, collect all actions a_t where b_t is in cluster k, and set $a(k)$ to the majority action.
3) **Transition assignment:** For each pair (k, o), consider all steps with b_t in cluster k and $o_t = o$, and look at the clusters of b_{t+1}. Set $\tau(k, o)$ to the most common

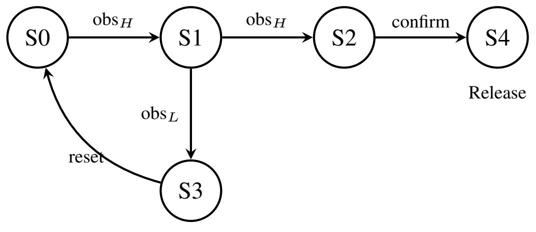


Fig. 2: Example 5-node FSC topology for adaptive drug delivery. Nodes represent internal evidence states; edges are labeled by biomarker observations and trigger state transitions leading to a release action.

next cluster; if no such transitions exist, use a self-loop $\tau(k, o) = k$.

Clustering is performed with standard k-means (Euclidean distance) initialized from random centroids. For the chronic-disease POMDP, we typically use modest controller sizes ($K = 4$–5) to balance expressivity and interpretability; the 5-node FSC in Fig. 2 illustrates a representative topology.

Fig. 2 illustrates an example 5-node FSC topology for adaptive drug delivery in an inflammatory disease domain: nodes encode internal "evidence" about persistent high inflammation, and transitions labeled by high/low observations and "reset" reflect accumulating evidence and occasional resets, terminating in a release state.

D. Compilation to DSD circuits

Given an FSC, we first compile it to a formal chemical-reaction network that explicitly represents controller nodes and observation pulses. For K nodes and $|\mathcal{O}|$ observations, we introduce signal species

- node species $Z_0, \ldots, Z_{K-1}$ corresponding to controller memory states, and
- observation species $O_0, \ldots, O_{|\mathcal{O}|-1}$ representing biomarker symbols delivered as pulses.

For each FSC transition $k \xrightarrow{o} k'$, we add a reaction $Z_k + O_o \to Z_{k'}$ to the CRN; optionally, we can add readout reactions $Z_k \to Z_k + A_{a(k)}$ to expose the action $a(k)$ at node k. This CRN implements the same discrete-time Mealy semantics as the FSC when run under pulsed observations.

To obtain a DSD implementation, we pass this controller CRN to Nuskell [12], using a standard translation scheme (e.g., `srinivas2015.ts`) to generate a domain-level DSD design in Peppercorn's `.pil` language. Nuskell introduces an internal domain vocabulary and gate complexes whose effective behavior realizes the input reactions, and it annotates the resulting `.pil` with the original CRN. We then feed the `.pil` file to Peppercorn, which enumerates the implied strand-displacement reaction network and associates approximate rate constants. In this workflow, our custom contribution is the systematic construction of the controller CRN from the FSC; the choice of gate motif and low-level domain assignment is delegated to Nuskell's translation scheme.

E. Topology-aware RL sequence optimization

Assigning concrete nucleotide sequences to the domains in a DSD circuit is a high-dimensional, constrained optimization problem. Traditional approaches (e.g., NUPACK design) optimize thermodynamics but do not explicitly optimize kinetics or leak in the context of a specific reaction topology. To close this loop, we define a topology-aware RL "gym" whose environment state encodes both sequence and circuit-level performance.

1) RL environment: state, action, reward: For a given DSD topology (hairpin or FSC-derived circuit), we define:

- **State:** one-hot encoded nucleotide sequences for all domains, together with scalar features derived from structure and topology. Concretely, the environment maintains a dictionary containing: (i) a tensor of shape $(\text{\#domains}, \text{max length}, 4)$ encoding the current sequences, (ii) NUPACK-derived features such as mean minimum free energy (MFE) across strands and a topology-aware crosstalk score that penalizes strong binding between strands that are not listed as reactants in any intended reaction, (iii) per-domain GC content, and (iv) bookkeeping features such as the remaining mutation budget. For neural policies, this dictionary is flattened into a fixed-length vector; for the hairpin benchmark this yields a 69-dimensional observation. When Peppercorn kinetics are used, we additionally attach coarse summaries of the enumerated reaction network (e.g., aggregate intended and leak rates), but the mapping between Peppercorn's symbolic complexes and the abstract domains and strands remains approximate.
- **Action:** a discrete mutation of the form "change base at position j in domain i to nucleotide $b \in \{\text{A,C,G,T}\}$". Internally, actions factor as $(\text{domain index}, \text{position index}, \text{base})$ in a product action space; for compatibility with standard PPO implementations we flatten this to a single discrete index. For the hairpin environment, this results in 64 possible actions; for larger circuits, the action space scales with the number and length of domains.
- **Reward:** a scalar combining multiple objectives linked to the circuit topology:
  - a *structure term* that rewards agreement between predicted structures and topology-specified targets when available;
  - a *crosstalk term* that penalizes strong unintended binding between strand pairs that are not reactants in any intended reaction, with thresholds taken from circuit-level constraints;
  - a *kinetics term* that favors large aggregate intended reaction rates relative to a minimum acceptable rate;
  - a *leak term* that penalizes large leak rates relative to a maximum acceptable leak threshold;
  - regularizers on GC content (encouraging values near 50%) and thermodynamic stability (more negative MFE).

In implementation these contributions are combined via simple piecewise-linear or logarithmic heuristics rather than a single closed-form expression, but they instantiate the qualitative form

$$r \approx w_{\text{struct}}(\text{structure}) - w_{\text{xtalk}}(\text{crosstalk}) + w_{\text{int}} \log k_{\text{int}} - w_{\text{leak}} \log k_{\text{leak}}, \tag{7}$$

with soft penalties for extreme GC content and unstable or overly metastable designs.

At each step of an episode, the agent proposes a mutation; the environment updates the sequence, calls NUPACK and Peppercorn to recompute features, and returns a reward. Episodes consist of a fixed number of mutations (e.g., $\sim$10); the goal is to maximize cumulative reward.

2) Policy architecture and training: We implement a 21.5K-parameter actor–critic network with:

- a 69-dimensional input,
- two hidden layers of size 64 with nonlinearities,
- a policy head outputting action logits over 64 actions, and
- a value head predicting state value.

We train this network using Proximal Policy Optimization (PPO) [13] over 100K environment steps, with rollout length 2048 and minibatch size 64, as implemented in Stable-Baselines3 [14]. PPO's clipped objective and value loss stabilize training while allowing the policy to explore sequence space.

V. Experiments and Discussion

We report experiments along two axes: (i) FSC approximation quality and interpretability, and (ii) RL sequence optimization performance on a benchmark hairpin circuit and the planned FSC-derived circuit.

A. FSC vs. belief-based policy

We simulate multiple episodes in the disease environment under both the belief-based policy $\pi(b)$ and the FSC-induced policy π_{FSC} (which chooses actions according to the node and updates nodes online via the same clustering mapping). Our primary metric is the average return over a fixed horizon:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{H-1} r_t\right]. \tag{8}$$

For moderate controller sizes (K = 4–5), π_{FSC} preserves most of the return of $\pi(b)$; the performance gap reflects approximation error from clustering and finite memory. In our experiments, we average over 200 episodes of length $H = 50$ with independent initial states and noise realizations for each policy.

The extracted FSCs are interpretable: nodes associated with beliefs concentrated on high disease and low toxicity states tend to choose aggressive dosing, while nodes associated with high toxicity select conservative actions. Transitions driven by "bad toxicity" observations move the controller into cautionary nodes; repeated "high inflammation" observations move it toward nodes closer to a release decision.

B. Controller CRN vs. DSD implementation

To verify that the DSD implementation produced by Nuskell and Peppercorn faithfully realizes the abstract controller dynamics, we compare trajectories in the controller CRN and the enumerated DSD reaction network. From the controller CRN we parse the node species $Z_0, \ldots, Z_{K-1}$ and reactions $Z_k + O_o \to Z_{k'}$; from Peppercorn's output we extract the corresponding signal species (again including Z_k and O_o) and the full set of mass-action reactions involving them and auxiliary fuel species. We initialize both systems with all mass on Z_0 and no observations, then integrate deterministic mass-action ODEs forward in time using a fixed-step Runge–Kutta method.

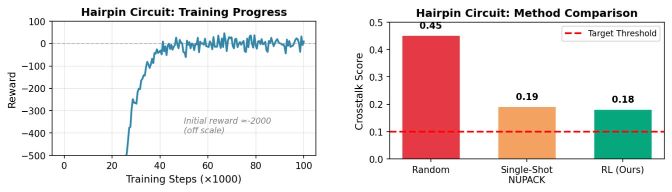
We consider both a baseline run with no observations and a run in which a pulse of observation species (e.g., O_0) is injected at time t_{obs}. At each time point, we aggregate concentrations of the node species to obtain a vector of "state scores," normalize to unit sum, and interpret the result as a belief-like vector over controller nodes. The normalized trajectories from the controller CRN and the DSD reaction network qualitatively agree: mass stays concentrated on Z_0 in the absence of observations, while an O_0 pulse shifts mass toward the same downstream nodes in both models. This supports the claim that the Nuskell/Peppercorn DSD realization preserves the intended finite-state controller dynamics at the level of node occupancy.

C. RL gym validation: hairpin benchmark

Before applying RL to FSC-derived circuits, we validate our topology-aware gym on a simple 2-domain hairpin: domain `d1` (8 nt) and `d2` (6 nt) forming target structure `((((....))))`. The goal is to find sequences that fold into this hairpin with favorable thermodynamics and no spurious alternatives.

The RL environment state includes the sequences for `d1` and `d2` and NUPACK-derived features (MFE, structural similarity, GC content, homopolymers). We train the 21.5K-parameter actor–critic network via PPO for 100K steps. In a representative run, episodes initially last only a single mutation with average reward $\sim$1.5 (essentially random proposals); by 50K–100K steps, the agent executes $\sim$10 mutations per episode and attains average reward around 6.0, indicating more deliberate mutation sequences and improved designs. Training this 100K-step run takes on the order of a minute on a single CPU core (roughly 1.8K environment steps per second). Fig. 3 shows training curves.

Qualitatively, the learned policy discovers known design heuristics: avoid long homopolymers (AAAA/GGGG), balance GC content near 50%, and drive the MFE below -10 kcal/mol. In particular, starting from random sequences with MFE typically in the range -2 to -5 kcal/mol, optimization often produces designs with MFE between -8 and -10 kcal/mol and near-zero crosstalk for non-interacting strands. Quantitatively, RL achieves comparable structural quality to NUPACK's native design on this simple benchmark. This supports the viability of using RL with NUPACK as an



(a) Training reward. (b) RL vs. NUPACK.

Fig. 3: Hairpin RL gym validation. Left: reward improves from $\sim$1.5 at initialization to peaks above 20 within the first 20K steps before settling near 6.0 as the policy converges. Right: RL-designed sequences achieve comparable quality to single-shot NUPACK design on this benchmark.

oracle and motivates applying it to more complex FSC-derived circuits where no prior heuristic exists.

D. *Toward FSC-POMDP circuit sequence design*

For the 5-node FSC topology in Fig. 2, we define a DSD circuit with:

- 7 distinct domains (transition toeholds and branch-migration regions),
- 6 strands (5 state strands plus one output signal strand),
- 4 intended reactions (state transitions triggered by observations).

We are in the process of specifying these complexes in dot-parens notation and instantiating an RL environment where the state includes NUPACK structural features and Peppercorn-estimated rates for intended transitions (e.g., $k_f > 10^{-3}$ s^{-1}) and leaks (e.g., $k_{\text{leak}} < 10^{-6}$ s^{-1}), computed in a topology-aware manner using the underlying FSC-derived circuit graph. A practical challenge is that Peppercorn's reaction enumeration introduces a large set of auxiliary complexes and species with autogenerated names; mapping these symbolic artifacts back to our abstract domains, strands, and intended reactions is nontrivial and currently relies on coarse summaries such as mean intended rate and worst-case leak rather than per-transition kinetics. Similarly, NUPACK and Peppercorn operate at different levels of coarse graining, with distinct assumptions about concentrations and rate scaling, so kinetic features in the RL state inevitably approximate rather than exactly match the effective in vitro behavior. The same PPO-based agent architecture will then be tasked with discovering sequences that realize the FSC transitions with favorable kinetics and minimal crosstalk, and improving this mapping between enumerated reaction networks, domain-level designs, and sequence-level constraints remains an important limitation and target for future work.

E. *Limitations and discussion*

Our study has several limitations. The disease model is stylized and does not capture the full complexity of inflammatory diseases. The POMDP solution is approximate and does not fully account for information-gathering actions. The FSC extraction is based on k-means and majority voting; more principled FSC optimization algorithms exist. The DSD gate motif is simple and not yet experimentally optimized. On the RL side, the hairpin benchmark is relatively easy; FSC-derived circuits will be more challenging due to larger state and action spaces and more complex energy landscapes. In addition, while our RL environment is topology-aware in that reward components and structural features are derived from a symbolic circuit description, the bridge between that description and the detailed reaction networks produced by tools like Peppercorn is imperfect: many intermediate complexes cannot be cleanly mapped back to individual FSC transitions, and kinetic summaries in the RL state necessarily compress this complexity. Finally, our sequence-generation policy currently relies on local point mutations and simple priors over bases; developing richer sequence moves and tighter integration between NUPACK- and Peppercorn-based evaluations is an important direction for improving design quality.

Despite these limitations, the combination of POMDP-derived FSCs, DSD compilation, and topology-aware RL represents, to our knowledge, the first systematic pipeline from high-level decision problems to molecular hardware. The RL gym framework is particularly appealing for idiosyncratic topologies (like FSC-POMDP controllers) where heuristic sequence design rules do not yet exist.

VI. Conclusion and Future Work

We presented a pipeline that starts from a POMDP model of chronic inflammatory disease management and ends with a domain-level DSD circuit implementing a finite-state approximation of an optimal dosing policy, along with a topology-aware RL environment for sequence design. Our main conclusions are:

- POMDP-derived dosing policies can be compressed into small finite-state controllers that preserve much of the original policy's performance and exhibit interpretable structure.
- These FSCs can be systematically compiled into DSD architectures, producing Peppercorn-compatible `.pil` files that reflect the controller topology.
- A topology-aware RL gym, coupled to NUPACK and Peppercorn, can learn sequence design heuristics and achieve competitive performance with established tools on a hairpin benchmark, and is positioned to tackle FSC-derived circuits where prior heuristics are lacking.

Future directions include: (i) adopting more realistic disease simulators or data-driven models, (ii) directly optimizing FSCs for POMDP performance (rather than post-hoc clustering), (iii) integrating experimentally validated DSD motifs and sequence-level constraints, (iv) scaling the RL gym to full FSC circuits and comparing RL-designed sequences against NUPACK-only baselines, and (v) ultimately, experimental validation of small FSC-based DNA controllers in vitro.

Author Contributions

Antonio Llano conceived the overall POMDP-to-DSD pipeline, implemented the rheumatoid-arthritis disease model and FSC extraction code, contributed to the design of the topology-aware RL gym, and led the writing of the paper. Shobhit Agarwal designed and implemented the topology-aware RL environment, including the DNA circuit gym, NUPACK/Peppercorn integration, and PPO-based training experiments. Ethan Goodhart developed the CRN and DSD simulation and comparison framework and contributed to the analysis of FSC vs. belief-based policies.

Acknowledgment

The authors thank collaborators and domain experts in rheumatology, molecular programming, and control theory for valuable discussions.

References

[1] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[2] M. Hauskrecht and H. Fraser, "Planning treatment of ischemic heart disease with partially observable Markov decision processes," *Artificial Intelligence in Medicine*, vol. 18, no. 3, pp. 221–244, 2000.

[3] O. Gottesman *et al.*, "Guidelines for reinforcement learning in healthcare," *Nature Medicine*, vol. 25, pp. 16–18, 2019.

[4] E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman, "Design and self-assembly of two-dimensional DNA crystals," *Nature*, vol. 394, pp. 539–544, 1998.

[5] L. Qian and E. Winfree, "Scaling up digital circuit computation with DNA strand displacement cascades," *Science*, vol. 332, no. 6034, pp. 1196–1201, 2011.

[6] J. N. Zadeh *et al.*, "NUPACK: analysis and design of nucleic acid systems," *Journal of Computational Chemistry*, vol. 32, no. 1, pp. 170–173, 2011.

[7] M. R. Lakin, D. P. Petrone, and A. Phillips, "Automated analysis of strand displacement systems using the Peppercorn tool," in *DNA Computing and Molecular Programming*, vol. 7433, Springer, 2012.

[8] K. T. Roybal *et al.*, "Precision tumor recognition by T cells with combinatorial antigen-sensing circuits," *Cell*, vol. 164, no. 4, pp. 770–779, 2016.

[9] C. C. Kloss *et al.*, "Control of CAR T cell activity: a dual receptor system," *Journal of Immunology*, vol. 191, no. 2, pp. 870–876, 2013.

[10] N. Meuleau *et al.*, "Learning finite-state controllers for partially observable environments," in *UAI*, 1999, pp. 427–436.

[11] C. Angermueller *et al.*, "Augmenting biological designs with deep learning," *Cell Systems*, vol. 10, no. 6, pp. 523–537, 2020.

[12] S. Badelt, S. W. Shin, R. F. Johnson, Q. Dong, C. Thachuk, and E. Winfree, "A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities," in *DNA Computing and Molecular Programming (DNA23)*, 2017, pp. 232–248.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv preprint arXiv:1707.06347, 2017.

[14] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.