Molecular Programming Models

Stable computation in the CRN model

Stochastic CRNs

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: n_1 is odd? I.e., $n_1 = 1 \bmod 2$

Reactions:

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: n_1 is odd? I.e., $n_1 = 1 \bmod 2$

Reactions:

$L_0 + X_1 \rightarrow L_1$ // even so far, then X_1 found, switch to odd

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: n_1 is odd? I.e., $n_1 = 1 \bmod 2$

Reactions:

$L_0 + X_1 \rightarrow L_1$ // even so far, then X_1 found, switch to odd
$L_1 + X_1 \rightarrow L_0$ // odd so far, then X_1 found, switch to even

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: n_1 is odd? I.e., $n_1 = 1 \bmod 2$

Reactions:

$L_0 + X_1 \rightarrow L_1$ // even so far, then X_1 found, switch to odd
$L_1 + X_1 \rightarrow L_0$ // odd so far, then X_1 found, switch to even

L_1 denotes that the answer is "yes", L_0 denotes "no"

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: n_1 is odd? I.e., $n_1 = 1 \bmod 2$

Reactions:

$L_0 + X_1 \rightarrow L_1$ // even so far, then X_1 found, switch to odd
$L_1 + X_1 \rightarrow L_0$ // odd so far, then X_1 found, switch to even

CRNs: Predicate computation

We can make our CRN leaderless (no context)

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)

~~plus a "leader" molecule L_0~~

Predicate: n_1 is odd? I.e., $n_1 = 1 \bmod 2$

Reactions:

$L_0 + X_1 \rightarrow L_1$ // even so far, then X_1 found, switch to odd

$L_1 + X_1 \rightarrow L_0$ // odd so far, then X_1 found, switch to even

CRNs: Predicate computation

We can make our CRN leaderless (no context)

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
~~plus a "leader" molecule L_0~~

Predicate: n_1 is odd? I.e., n_1 = 1 mod 2

Reactions:

$L_0 + X_1 \rightarrow L_1$ // even so far, then X_1 found, switch to odd
$L_1 + X_1 \rightarrow L_0$ // odd so far, then X_1 found, switch to even
$X_1 \rightarrow L_1$ // one X_1 found, so odd

CRNs: Predicate computation

We can make our CRN leaderless (no context)

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
~~plus a "leader" molecule L_0~~

Predicate: n_1 is odd? I.e., $n_1 = 1 \bmod 2$

Reactions:

$L_0 + X_1 \rightarrow L_1$ // even so far, then X_1 found, switch to odd

$L_1 + X_1 \rightarrow L_0$ // odd so far, then X_1 found, switch to even

$X_1 \rightarrow L_1$ // one X_1 found, so odd

$L_0 + L_1 \rightarrow L_1$ // even plus odd is odd

$L_1 + L_1 \rightarrow L_0$ // odd plus odd is even

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_2 < n_1 - 2$?

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_2 < n_1 - 2$?

Reactions:

$L_0 + X_1 + X_1 + X_1 \rightarrow L_Y$ // base case: 0 (X_2's) < 3 (X_1's) -2

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_2 < n_1 - 2$?

Reactions:

$L_0 + X_1 + X_1 + X_1 \rightarrow L_Y$ // base case: 0 (X_2's) < 3 (X_1's) -2

$L_Y + X_2$

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_2 < n_1 - 2$?

Reactions:

$L_0 + X_1 + X_1 + X_1 \rightarrow L_Y$ // base case: 0 (X_2's) < 3 (X_1's) -2
$L_Y + X_2 \rightarrow L_N$ // extra X_2, so predicate is false
$L_N + X_1 \rightarrow L_Y$ // extra X_1, predicate is true again

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_2 < n_1 - 2$?

Reactions:

$L_0 + X_1 + X_1 + X_1 \rightarrow L_Y$ // base case: 0 (X_2's) < 3 (X_1's) -2

$L_Y + X_2 \rightarrow L_N$ // extra X_2, so predicate is false

$L_N + X_1 \rightarrow L_Y$ // extra X_1, predicate is true again

L_Y denotes "yes"; L_0 and L_N denote "no"

CRNs: Predicate computation

- Let C = ($\mathbf{\Lambda}$,R) be a CRN with input species $X_1, \ldots, X_k$
- Some species of $\mathbf{\Lambda}$ are *yes-voters*
- Some species of $\mathbf{\Lambda}$ are *no-voters*
- An *input configuration* C_{init} is one in which the counts of all but the input species is 0
- We'll denote $C_{init}(X_i)$ by n_i (initial count of species X_i), and let $n = n_{1,} + n_2 + \ldots + n_k$

CRNs: Predicate computation

- For a configuration c, let $\phi(c)$ be
  - 1 if c contains yes-voters but no no-voters
  - 0 if c contains no-voters but no yes-voters
  - undefined otherwise
- A *configuration* o is *output stable* if $\phi(o)$ is defined, and for all c such that $o \rightarrow c$, $\phi(o) = \phi(c)$

CRNs: Predicate computation

- Let ψ: $\mathbb{N}^k \rightarrow$ {0,1} be a predicate
- We say that C computes ψ if for all input configurations C_{init}, $C_{init} \rightarrow$ c implies that c $\rightarrow$ o, where o is output stable and $\phi(o) = \psi(n_1, n_2, \ldots, n_k)$
- Equivalently, C decides set A where $(n_1, n_2, \ldots, n_k)$ is in A iff $\psi(n_1, n_2, \ldots, n_k) = 1$

CRNs: Predicate computation

- Let s be a configuration in which only non-input species may have counts > 0

- A *CRN decider with initial context s* is just like a CRN decider, except that the initial configuration is C_{init} + s, where C_{init} is an input configuration

CRNs: Predicate computation

Claim: Mod sets are stably decidable by CRNs

Mod set *X*: for some constants $a_1, a_2, \ldots, a_d \in \mathbb{Z}$, $b, c \in \mathbb{N}$

$$X = \left\{ \mathbf{x} \in \mathbb{N}^d \;\middle|\; \sum_{i=1}^{d} a_i \mathbf{x}(i) \equiv b \mod c \right\}$$

Proof by example in the following slides.

CRNs: Predicate computation

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 = 1 \bmod 2$

Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_0$$

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 = 1 \bmod 2$

Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_0$$

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2) plus a "leader" molecule L_0

Predicate: $n_1 = 1 \bmod 2$

Predicate: $n_1 = 2 \bmod 3$

Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_0$$

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 = 1 \bmod 2$

Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_0$$

Predicate: $n_1 = 2 \bmod 3$

Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_2$$
$$L_2 + X_1 \rightarrow L_0$$

L_2 means "yes",
L_0 and L_1 mean "no"

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 = 2 \bmod 3$

Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_2$$
$$L_2 + X_1 \rightarrow L_0$$

L_2 means "yes",
L_0 and L_1 mean "no"

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 = 2 \bmod 3$

Predicate: $n_1 + 4n_2 = 2 \bmod 3$

Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_2$$
$$L_2 + X_1 \rightarrow L_0$$

L_2 means "yes",
L_0 and L_1 mean "no"

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 = 2 \bmod 3$

Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_2$$
$$L_2 + X_1 \rightarrow L_0$$

L_2 means "yes",
L_0 and L_1 mean "no"

Predicate: $n_1 + 4n_2 = 2 \bmod 3$

Reactions:

$$X_1 \rightarrow Z_1$$
$$X_2 \rightarrow Z_1 + Z_1 + Z_1 + Z_1$$

Replace X_1 by Z_1 in the CRN for predicate "$n_1 = 2 \bmod 3$"

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 + 4n_2 = 2 \bmod 3$

Reactions:

$X_1 \rightarrow Z_1$
$X_2 \rightarrow Z_1 + Z_1 + Z_1 + Z_1$

$L_0 + Z_1 \rightarrow L_1$
$L_1 + Z_1 \rightarrow L_2$
$L_2 + Z_1 \rightarrow L_0$

L_2 means "yes",
L_0 and L_1 mean "no"

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 - 4n_2 = 2 \bmod 3$

substract instead of add?

Reactions:

$X_1 \rightarrow Z_1$
$X_2 \rightarrow Z_1 + Z_1 + Z_1 + Z_1$

$L_0 + Z_1 \rightarrow L_1$
$L_1 + Z_1 \rightarrow L_2$
$L_2 + Z_1 \rightarrow L_0$

L_2 means "yes",
L_0 and L_1 mean "no"

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 - 4n_2 = 2 \bmod 3$

substract instead of add?

Reactions:

$X_1 \rightarrow Z_1$

$X_2 \rightarrow W_1 + W_1 + W_1 + W_1$

$L_0 + Z_1 \rightarrow L_1$

$L_1 + Z_1 \rightarrow L_2$

$L_2 + Z_1 \rightarrow L_0$

L_2 means "yes",
L_0 and L_1 mean "no"

CRNs: Predicate computation

More general mod functions

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_1 - 4n_2 = 2 \bmod 3$

substract instead of add?

Reactions:

$X_1 \rightarrow Z_1$

$X_2 \rightarrow W_1 + W_1 + W_1 + W_1$

L_2 means "yes",
L_0 and L_1 mean "no"

$L_0 + Z_1 \rightarrow L_1$

$L_1 + Z_1 \rightarrow L_2$

$L_2 + Z_1 \rightarrow L_0$

$L_0 + W_1 \rightarrow L_2$

$L_1 + W_1 \rightarrow L_0$

$L_2 + W_1 \rightarrow L_1$

CRNs: Predicate computation

Claim: Mod sets are stably decidable by CRNs

Mod set *X*: for some constants $a_1, a_2, \ldots, a_k \in \mathbb{Z}$, b, c $\in \mathbb{N}$

$$X = \{ n \in \mathbb{N}^k \mid a_1.n_1 + a_2.n_2 + \ldots + a_k.n_k = b \bmod c\}$$

CRNs: Predicate computation

Claim: Threshold sets are stably decidable by CRNs

Threshold set *X*: for some constants b, $a_1, a_2, \ldots, a_k \in \mathbb{Z}$,

$$X = \{ n \in \mathbb{N}^k \mid a_1.n_1 + a_2.n_2 + \ldots + a_k.n_k < b \}$$

CRNs: Predicate computation

Threshold set example (not too hard to generalize)

Initial state: (n_1, n_2) (represented by counts of X_1, X_2)
plus a "leader" molecule L_0

Predicate: $n_2 < n_1 - 2$?

Reactions:

$L_0 + X_1 + X_1 + X_1 \rightarrow L_Y$ // base case: 0 (X_2's) < 3 (X_1's) -2

$L_Y + X_2 \rightarrow L_N$ // extra X_2, so predicate is false

$L_N + X_1 \rightarrow L_Y$ // extra X_1, predicate is true again

L_Y denotes "yes"; L_0 and L_N denote "no"

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.
What CRN computes the complement of P_1?

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.
What CRN computes the complement of P_1?

Example: P_1 is the predicate "$n_1 = 1 \bmod 2$"
Reactions:

$$L_0 + X_1 \rightarrow L_1$$
$$L_1 + X_1 \rightarrow L_0$$

L_1 denotes "yes", L_0 denotes "no"

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively. What CRN computes the complement of P_1?

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.
What CRN computes the complement of P_1?

The CRN obtained by swapping "yes" and "no" species of CRN stably C_1 computes the complement of P_1

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

- For each "no" species N_1 of C_1, add reactions:

  $N_1 + YN \rightarrow N_1 + NN$ and $N_1 + YY \rightarrow N_1 + NY$

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

- For each "no" species N_1 of C_1, add reactions:
  $N_1 + YN \rightarrow N_1 + NN$ and $N_1 + YY \rightarrow N_1 + NY$
- For each "no" species N_2 of C_2, add reactions:

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

- For each "no" species N_1 of C_1, add reactions:
  $N_1 + YN \rightarrow N_1 + NN$ and $N_1 + YY \rightarrow N_1 + NY$
- For each "no" species N_2 of C_2, add reactions:
  $N_2 + NY \rightarrow N_2 + NN$ and $N_2 + YY \rightarrow N_2 + YN$

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

- For each “yes” species Y_1 of C_1, add reactions:

- For each “yes” species Y_2 of C_2, add reactions:

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

- For each "yes" species Y_1 of C_1, add reactions:
  $Y_1 + NY \rightarrow Y_1 + YY$ and $Y_1 + NN \rightarrow N_1 + YN$
- For each "yes" species Y_2 of C_2, add reactions:

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

- For each "yes" species Y_1 of C_1, add reactions:
  $Y_1 + NY \rightarrow Y_1 + YY$ and $Y_1 + NN \rightarrow N_1 + YN$
- For each "yes" species Y_2 of C_2, add reactions:
  $Y_2 + YN \rightarrow Y_2 + YY$ and $Y_2 + NN \rightarrow N_2 + NY$

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.
We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

- NN is the "no" voter
- NY, YN, YY are "yes" voters

CRNs: Predicate computation

Claim: Stably computable predicates are closed under union, intersection and complement

Let C_1 and C_2 compute predicates P_1 and P_2, respectively.

We want a CRN that computes the union of P_1 and P_2

Introduce four new species: NN, NY, YN, YY

- NN is the “no” voter
- NY, YN, YY are “yes” voters

One of the species, say NN, is in the initial context

CRNs: Predicate computation

Summary so far:

- Mod predicates are stably computable
- Threshold predicates are stably computable
- Stably computable predicates are closed under union, intersection and complement

CRNs: Predicate computation

Summary so far:

- Mod predicates are stably computable
- Threshold predicates are stably computable
- Stably computable predicates are closed under union, intersection and complement

Definition: *Semilinear predicates* are finite unions, intersections and complements of threshold sets and mod sets

CRNs: Predicate computation

Summary so far:

- Mod predicates are stably computable
- Threshold predicates are stably computable
- Stably computable predicates are closed under union, intersection and complement

Definition: *Semilinear predicates* are finite unions, intersections and complements of threshold sets and mod sets

Claim: Predicates stably computable by CRNs are exactly the semilinear predicates

CRNs: Stochastic (kinetic) model

CRNs: Stochastic (kinetic) model

- So far: we’ve developed intuition about what's “stably” decidable (and undecidable)
- Next, we’d like to distinguish between predicates that can, or cannot, be decided *quickly*
- For this we’ll use a stochastic model of reaction rates

CRNs: Stochastic (kinetic) model

- Fix a configuration c, let v be the volume of the system
- The *propensity* of reaction (r,p) is given by:

reaction type (r,p)	propensity(r,p)
$X_i \rightarrow \ldots$	$\lvert X_i \rvert$
$X_i + X_j \rightarrow \ldots$	$\lvert X_i \rvert . \lvert X_i \rvert / v$
$X_i + X_i \rightarrow \ldots$	$\lvert X_i \rvert . \lvert X_i -1 \rvert / 2v$

- The probability that (r,p) occurs next is
  $\text{prop}(r,p) / \sum_{(r',p')} \text{prop}(r',p')$
- The *expected time* for a reaction is $1/\sum_{(r',p')} \text{prop}(r',p')$

Summary

- CRNs can stably decide exactly the semilinear predicates
- A stochastic model is used to model reaction rates and expected times, which we'll need to define time complexity of CRN "algorithms"