Molecular Programming Models

Expected time to stably compute predicates

CRNs with probability of error

Based on notes by Dave Doty

CRNs: Expected runtime analysis

Assumptions for stable computation

- Total number of inputs is $n = n_1 + n_2 + \ldots + n_k$
- The volume is (proportional to) n
- For now: throughout a computation, the total number of species is O(n)

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X \rvert$
$X + X' \rightarrow \ldots$	$\lvert X \rvert . \lvert X' \rvert / v$

current count of species X

$X \neq X'$

Expected time for a reaction: $1/\sum_{(r,p)} \text{prop}(r,p)$

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
X → …	|X|
X + X' → …	|X|. |X'| / v

Expected time for a reaction: $1/\sum_{(r,p)} \text{prop}(r,p)$

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X\rvert$
$X + X' \rightarrow \ldots$	$\lvert X\rvert . \lvert X'\rvert / v$

Expected time for a reaction: $1/\sum_{(r,p)} prop(r,p)$

"No communication" example: multiply by 2: X --> 2Y:

- Let Ti be the expected time for a reaction, when $|X| = i$

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X \rvert$
$X + X' \rightarrow \ldots$	$\lvert X \rvert . \lvert X' \rvert / v$

Expected time for a reaction: $1/\sum_{(r,p)} \text{prop}(r,p)$

"No communication" example: multiply by 2: X --> 2Y:

- Let Ti be the expected time for a reaction, when $|X| = i$
- Then $Ti = 1/i$

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\|X\|$
$X + X' \rightarrow \ldots$	$\|X\|.\ \|X'\| / v$

Expected time for a reaction: $1/\sum_{(r,p)} \text{prop}(r,p)$

"No communication" example: multiply by 2: X --> 2Y:

- Let Ti be the expected time for a reaction, when $|X| = i$
- Then $Ti = 1/i$
- Expected time for all X's to react is

  $\sum_i Ti = \sum_i 1/i = \Theta(\log n)$

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X \rvert$
$X + X' \rightarrow \ldots$	$\lvert X \rvert . \lvert X' \rvert / v$

Expected time for a reaction: $1/\sum_{(r,p)} \text{prop}(r,p)$

"Pairing off" example: Min(n1,n2): X1 + X2 --> Y:

- Suppose that n1 ≥ n2

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X \rvert$
$X + X' \rightarrow \ldots$	$\lvert X \rvert . \lvert X' \rvert / v$

Expected time for a reaction: $1/\sum_{(r,p)} \text{prop}(r,p)$

"Pairing off" example: Min(n1,n2): X1 + X2 --> Y:

- Suppose that n1 ≥ n2
- Let Ti be exp. time for a reaction when |X2| = i

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X \rvert$
$X + X' \rightarrow \ldots$	$\lvert X \rvert . \lvert X' \rvert / v$

Expected time for a reaction: $1/\sum_{(r,p)} prop(r,p)$

"Pairing off" example: Min(n1,n2): X1 + X2 --> Y:

- Suppose that $n1 \geq n2$
- Let Ti be exp. time for a reaction when $|X2| = i$
- Then $Ti = n/(i|X1|) \leq n/i^2$

CRNs: Expected runtime analysis

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X \rvert$
$X + X' \rightarrow \ldots$	$\lvert X \rvert . \lvert X' \rvert / v$

Expected time for a reaction: $1/\sum_{(r,p)} \text{prop}(r,p)$

"Pairing off" example: Min(n1,n2): X1 + X2 --> Y:

- Suppose that $n1 \geq n2$
- Let Ti be exp. time for a reaction when $\lvert X2 \rvert = i$
- Then $Ti = n/(i\lvert X1 \rvert) \leq n/i^2$
- Expected time for all reactions is

$$\sum_i Ti = n \sum_i 1/i^2 = O(n)$$

CRNs: Expected runtime analysis

- $n_1 - n_2$ (assume that $n_1 \geq n_2$): $X1 \rightarrow Y$

  $X2 + Y \rightarrow \emptyset$

CRNs: Expected runtime analysis

- $n_1 - n_2$ (assume that $n_1 \geq n_2$): $X1 \rightarrow Y$

  $X2 + Y \rightarrow \emptyset$

- To get an upper bound on the expected time, assume that the second reaction doesn't start until the first completes

CRNs: Expected runtime analysis

- $n_1 - n_2$ (assume that $n_1 \geq n_2$): $X1 \rightarrow Y$
  $X2 + Y \rightarrow \emptyset$

- To get an upper bound on the expected time, assume that the second reaction doesn't start until the first completes
- Then sum the expected time for the first to complete ("no communication") plus the expected time for the second to complete given that the first has completed ("pairing off"): $O(\log n) + O(n) = O(n)$

CRNs: Expected runtime analysis

Exercise:

- $\max(n_1,n_2)$:

$$
\begin{aligned}
X1 &\rightarrow Y + Z1 \\
X2 &\rightarrow Y + Z2 \\
Z1 + Z2 &\rightarrow K \\
K + Y &\rightarrow \emptyset
\end{aligned}
$$

CRNs: Expected runtime analysis

Exercise:

- $\max(n_1,n_2)$:

$$
\begin{aligned}
X1 &\rightarrow Y + Z1 \\
X2 &\rightarrow Y + Z2 \\
Z1 + Z2 &\rightarrow K \\
K + Y &\rightarrow \emptyset
\end{aligned}
$$

  - Assume that the first two reactions complete before the third starts, and that the third completes before the fourth starts

CRNs: Expected runtime analysis

Exercise:

- $\max(n_1,n_2)$:

$$
\begin{aligned}
X1 &\rightarrow Y + Z1 \\
X2 &\rightarrow Y + Z2 \\
Z1 + Z2 &\rightarrow K \\
K + Y &\rightarrow \emptyset
\end{aligned}
$$

  - Assume that the first two reactions complete before the third starts, and that the third completes before the fourth starts
  - Then we have “no communication” followed by “pairing off”, and another “pairing off”.
  - Total expected time is O(n)

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

- The reactions must alternate
- Expected time for first, when i copies of X2 left, is …?

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X \rvert$
$X + X' \rightarrow \ldots$	$\lvert X \rvert . \lvert X' \rvert / v$

Expected time for a reaction: $1/\sum_{(r,p)} \text{prop}(r,p)$

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

- The reactions must alternate
- Expected time for first, when i copies of X2 left, is n/i

reaction type (r,p)	propensity(r,p)
$X \rightarrow \ldots$	$\lvert X \rvert$
$X + X' \rightarrow \ldots$	$\lvert X \rvert . \lvert X' \rvert / v$

Expected time for a reaction: $1/\sum_{(r,p)} \mathrm{prop}(r,p)$

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

- The reactions must alternate
- Expected time for first, when i copies of X2 left, is n/i

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

- The reactions must alternate
- Expected time for first, when i copies of X2 left, is n/i
- Expected time for second, when i copies of X1 left, is …

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

- The reactions must alternate
- Expected time for first, when i copies of X2 left, is n/i
- Expected time for second, when i copies of X1 left, is n/i

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

- The reactions must alternate
- Expected time for first, when i copies of X2 left, is n/i
- Expected time for second, when i copies of X1 left, is n/i
- Total expected time is

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

- The reactions must alternate
- Expected time for first, when i copies of X2 left, is n/i
- Expected time for second, when i copies of X1 left, is …
- Total expected time is $2n \sum_i 1/i = O(n \log n)$

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

$$X1 + X2 \rightarrow \emptyset$$

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

$$X1 + X2 \rightarrow \emptyset$$

- Assume that the last reaction finishes before the first two start

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$
$$L_Y + X1 \rightarrow L_N$$
$$X1 + X2 \rightarrow \emptyset$$

- Assume that the last reaction finishes before the first two start
- The last completes in O(n) expected time ("pairing off”)

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$

$$L_Y + X1 \rightarrow L_N$$

$$X1 + X2 \rightarrow \emptyset$$

- Assume that the last reaction finishes before the first two start
- The last completes in O(n) expected time ("pairing off")
- Then, once one of the remaining two happens, a stable configuration is reached

CRNs: Expected runtime analysis

Exercise:

- $n_1 < n_2$? Initial context L_N

$$L_N + X2 \rightarrow L_Y$$
$$L_Y + X1 \rightarrow L_N$$
$$X1 + X2 \rightarrow \emptyset$$

- Assume that the last reaction finishes before the first two start
- The last completes in O(n) expected time ("pairing off")
- Then, once one of the remaining two happens, a stable configuration is reached
- So total expected time is O(n)

CRNs: Expected runtime analysis

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

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

- More generally, there is a CRN to stably compute Threshold in O(n) expected time

CRNs: Expected runtime analysis

Exercise: Is n_1 odd ? Initial context L_0

$$L_0 + X1 \rightarrow L_1$$

$$L_1 + X1 \rightarrow L_0 \;\; L$$

CRNs: Expected runtime analysis

Exercise: Is n_1 odd ? Initial context L_0

$$L_0 + X1 \rightarrow L_1$$
$$L_1 + X1 \rightarrow L_0 \; L$$

- The reactions must alternate
- Expected time for either, when i copies of X1 left, is n/i
- Total expected time is $2n \sum_i 1/i = O(n \log n)$
- Is there a faster CRN?

CRNs: Expected runtime analysis

Exercise: Is n_1 odd ? Initial context L_0

$$L_0 + X1 \rightarrow L_1$$
$$L_1 + X1 \rightarrow L_0 \; L$$
$$X1 + X1 \rightarrow \emptyset$$

- Assume that the last reaction finishes before the first two start
- The last completes in O(n) expected time ("pairing off)
- Then, in at most one more reaction (which takes O(n) expected time), a stable configuration is reached
- So total expected time is O(n)

CRNs: Expected runtime analysis

Exercise: Is n_1 odd ? Initial context L_0

$$L_0 + X1 \rightarrow L_1$$

$$L_1 + X1 \rightarrow L_0 \;\; L$$

$$X1 + X1 \rightarrow \emptyset$$

CRNs: Expected runtime analysis

Exercise: Is n_1 odd ? Leaderless

$L_0 + X_1 \rightarrow L_1$ // 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: Expected runtime analysis

Exercise: Is n_1 odd ? Leaderless

$L_0 + X_1 \rightarrow L_1$ // 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

- Unimolecular reaction completes in O(log n) exp. time
- The last two complete in O(n) exp. time ("pairing off)

CRNs: Expected runtime analysis

Exercise: Is n_1 odd ? Leaderless

$L_0 + X_1 \rightarrow L_1$ // 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

- Unimolecular reaction completes in O(log n) exp. time
- The last two complete in O(n) exp. time ("pairing off)
- Then no more X1's and just one L (either L_0 or L_1), so no more reactions
- Total expected time is O(n)

CRNs: Expected runtime analysis

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

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

- More generally, there is a CRN to stably compute Mod in O(n) expected time

CRNs: Expected runtime analysis

Claim: All semilinear predicates can be stably decided in O(n) expected time

Proof: Follows by analyzing CRNs for finite union, intersection, and complement, as well as threshold and mod sets.

CRNs: Expected runtime analysis

Claim: All semilinear predicates can be stably decided in O(n) expected time

Proof: Follows by analyzing CRNs for finite union, intersection, and complement, as well as threshold and mod sets.

Note that some semilinear predicates, e.g., “Multiply by 2”, can be stably decided in O(log n) expected time.

Other notions of CRN predicate computation

Committing CRNs

- A CRN that stably decides A is a *committing* CRN if, for any initial configuration, it is possible either to reach a “yes” configuration or a “no” configuration, but not both

- Intuitively, a committing CRN "knows" when it is done

Committing CRNs

- A CRN that stably decides A is a *committing* CRN if, for any initial configuration, it is possible either to reach a “yes” configuration or a “no” configuration, but not both
- Intuitively, a committing CRN "knows" when it is done
- Unfortunately, committing protocols can only decide the sets $A = \mathbb{N}^k$ and $A = \emptyset$ (the “constant” predicates)

CRNs with bounded error

- Let C be a CRN and x an input to C.
- Let Prob[C accepts x] be the probability that C reaches a "yes"-stable configuration on input x

- A CRN C *stably decides a predicate A with error probability* ε if
  - for all x in A, Prob[C accepts x] ≥ 1 – ε
  - for all x not in A, Prob[C accepts x] ≤ ε

Summary: CRN computational power

	committing	stable
Prob correct = 1	constant	semilinear
Bounded error	computable	

Summary: CRN computational power

	committing	stable
Prob correct = 1	constant	semilinear
Bounded error	computable	

To show how CRNs can decide any computable predicate (by a Turing machine), we'll introduce register (counter) machines

Register machines

A *register machine* is a finite sequence of instructions from the following set:

- accept
- reject
- goto j // go to the jth instruction
- inc r_i // add one to counter Rj
- dec r_i , j // if $r_i > 0$, subtract one from r_i, otherwise
  // go to the jth instruction

The input n_1, …, n_k is the initial value of the first k counters.

Register machine examples

Predicate: Is n_1 odd?

Register machine:

1. dec n_1, 4
2. dec n_1, 5
3. goto 1
4.
5.

Register machine examples

Predicate: Is n_1 odd?

Register machine:

1. dec n_1, 4
2. dec n_1, 5
3. goto 1
4. reject
5. accept

Register machine examples

Predicate: Is $n_1 < n_2$?

Register machine:

1. dec n_2, 6
2. dec n_1, 4
3. goto 1
4.
5.
6.

Register machine examples

Predicate: Is $n_1 < n_2$?

Register machine:

1. dec n_2, 6
2. dec n_1, 4
3. goto 1
4.
5.
6. reject

Register machine examples

Predicate: Is $n_1 < n_2$?

Register machine:

1. dec n_2, 5
2. dec n_1, 4
3. goto 1
4. accept
5. reject

Register machine examples

Predicate: Is $n_1 < n_2$?

Register machine:

1. dec n_2, 5
2. dec n_1, 4
3. goto 1
4. accept
5. reject

Predicate: Is $n_1 = n_2$?

Register machine:

Register machine examples

Predicate: Is $n_1 < n_2$?

Register machine:

1. dec n_2, 6
2. dec n_1, 4
3. goto 1
4. dec n_2, 6
5. accept
6. reject

Predicate: Is $n_1 = n_2$?

Register machine:

1. dec n_2, 4
2. dec n_1,
3. goto 1

Register machine examples

Predicate: Is $n_1 < n_2$?

Register machine:

1. dec n_2, 6
2. dec n_1, 4
3. goto 1
4. dec n_2, 6
5. accept
6. reject

Predicate: Is $n_1 = n_2$?

Register machine:

1. dec n_2, 4
2. dec n_1,
3. goto 1
4. dec n_1, 6

Register machine examples

Predicate: Is $n_1 < n_2$?

Register machine:

1. dec n_2, 6
2. dec n_1, 4
3. goto 1
4. dec n_2, 6
5. accept
6. reject

Predicate: Is $n_1 = n_2$?

Register machine:

1. dec n_2, 4
2. dec n_1,
3. goto 1
4. dec n_1, 6
5. reject
6. accept

Register machine examples

Predicate: Is $n_1 < n_2$?

Register machine:

1. dec n_2, 6
2. dec n_1, 4
3. goto 1
4. dec n_2, 6
5. accept
6. reject

Predicate: Is $n_1 = n_2$?

Register machine:

1. dec n_2, 4
2. dec n_1, 5
3. goto 1
4. dec n_1, 6
5. reject
6. accept

Register machine examples

Predicate: Is $n_1^2 = n_2$?

Register machine:

copy $r_1 \rightarrow r_3$

$r_4 \leftarrow r_1 \times r_3$

$r_2 = r_4$?

Register machine examples

Handy subroutines

flush $r_1 \rightarrow r_2, r_3$:

// set r_2, r_3 to the initial value

// of r_1 and set r_1 to 0

1. dec r_1, 5
2. inc r_2
3. inc r_3
4. goto 1
5. …

Register machine examples

Handy subroutines

flush $r_1 \rightarrow r_2, r_3$:

// set r_2, r_3 to the initial value

// of r_1 and set r_1 to 0

1. dec r_1, 5
2. inc r_2
3. inc r_3
4. goto 1
5. …

flush $r_1 \rightarrow r_2$: similar, but no r_3

Register machine examples

Handy subroutines

flush $r_1 \rightarrow r_2, r_3$:

// set r_2, r_3 to the initial value

// of r_1 and set r_1 to 0

1. dec r_1, 5
2. inc r_2
3. inc r_3
4. goto 1
5. …

flush $r_1 \rightarrow r_2$: similar, but no r_3

copy $r_1 \rightarrow r_2$:

// copy r_1 to r_2

flush $r_1 \rightarrow r_2, r_3$

flush $r_3 \rightarrow r_1$

Register machine examples

Handy subroutines

$r_1 \leftarrow r_2 \times r_3$: // add r_2 times r_3 to r_1

Register machine examples

Handy subroutines

$r_1 \leftarrow r_2 \times r_3$: // add r_2 times r_3 to r_1

1. dec r_2, 7
2. copy $r_3 \rightarrow r_4$
3. dec r_4, 6
4. inc r_1
5. goto 3
6. goto 1
7. …

```
While r2 > 0
    dec r2
    While r3 > 0
        dec r3
        inc r1
```

Register machines can simulate Turing machines

- A stack machine can simulate a Turing machine
- A register machine can simulate a stack machine

Register machines can simulate Turing machines

- A stack machine can simulate a Turing machine
  - Store left part of TM tape, including tape head, on one stack
  - Store rest of TM tape on another stack
  - Simulate a TM transition via pops and pushes
- A register machine can simulate a stack machine

Register machines can simulate Turing machines

- A stack machine can simulate a Turing machine
  - Store left part of TM tape, including tape head, on one stack
  - Store rest of TM tape on another stack
  - Simulate a TM transition via pops and pushes

- A register machine can simulate a stack machine
  - Represent binary stack using a unary counter
  - Push “0” : double the counter
  - Push“1” : double plus increment
  - Pop: divide counter by 2; test whether even or odd to determine value at top of stack

CRNs can simulate register machines

- One species R_i per register
- Initial count of R_i is n_i if R_i is an input register, and is 0 otherwise
- One species L_i for each instruction number i
- Initial context is L_1
- Instructions:

`accept`	$L_i \to Y$
`reject`	$L_i \to N$
`goto` k	$L_i \to L_k$
`inc` r_j	$L_i \to L_{i+1} + R_j$
`dec` r_j, k	$L_i + R_j \to L_{i+1}$???

CRNs can simulate register machines

- One species R_i per register
- Initial count of R_i is n_i if R_i is an input register, and is 0 otherwise
- One species L_i for each instruction number i
- Initial context is L_1
- Instructions:

`accept`	$L_i \rightarrow Y$
`reject`	$L_i \rightarrow N$
`goto` k	$L_i \rightarrow L_k$
`inc` r_j	$L_i \rightarrow L_{i+1} + R_j$
`dec` r_j, k	$L_i + R_j \rightarrow L_{i+1}$ $L_i \rightarrow L_k$

CRNs can simulate register machines

dec r_j, k	$L_i + R_j \to L_{i+1}$ $L_i \to L_k$

Propensity is r_j/v

Propensity is 1

Probability of error at *each* decrement is

$$1/(1 + r_j/v) = v/(v + r_j)$$

We need to reduce the error!