Biochemical Reactions and Logic Computation

Jie-Hong Roland Jiang

(Introduction to EDA, 2013)

National Taiwan University

Biochemical Reactions and Biology

- Complex behaviors of a living organism originate from systems of biochemical reactions

- Engineering biochemical reactions may sharpen our understanding on how nature design living organisms (in contrast to how human design electronic systems)

Outline

- Compiling program control flows into biochemical reactions
- Beyond logic computation

Outline

- Compiling program control flows into biochemical reactions
  - Joint work with Chi-Yun Cheng, De-An Huang, Ruei-Yang Huang [ICCAD 2012]
- Beyond logic computation

Computational Biochemistry

- In living organisms, biochemical reactions carry out some form of “computations” which result in complex behaviors
- Biochemical reactions may be exploited for computation by combining a proper set of biochemical reactions
- Computation with biochemical reactions has potential applications in synthetic biology
  - In synthetic biology, known biochemical parts (DNA, mRNA, proteins, etc.) are assembled either naturally or artificially to realize desired functions

Previous Work

- Synthesizing molecular reactions has been pursued, e.g.,
  - Arithmetic operations
    - Fett *et al.* (2007, 2008)
  - Digital signal processing
    - Jiang *et al.* (2010)
  - Writing and compiling code into biochemistry
    - Shea *et al.* (2010), Senum *et al.* (2011)

Previous Work

- Still lack systematic methodology to construct complex program **control flows**
- Heavily rely on modularized reactions
- Assume reactions are of small quantities
  - Work under stochastic simulation but not ODE simulation

Our Focus

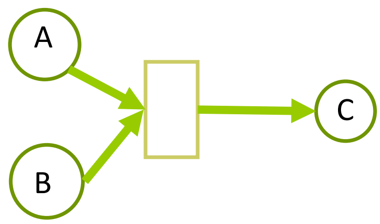
- Robustness
  - Improved reaction regulation
  - Enhanced fault tolerance
  - Valid under both stochastic and ODE simulations
- Optimality
  - Reduced number of reactions
  - Not limited to modularized reactions
- Systematic compilation methodology

Model of Computation

- Computation with biochemical reactions
  - Computation in terms of molecular quantities
  - Quantity changing rules are defined by reactions
- Example

$$A + B \rightarrow C$$

$$[C] = \min\{[A],[B]\}$$



Petri net representation

Reaction Model

- Classical chemical kinetic (CCK) model
  - ODE based simulation
    - (Empirical study shows our construction works for discrete stochastic simulation as well)
- Example:

$$\alpha A + \beta B \xrightarrow{k} \gamma C$$

$$-\frac{1}{\alpha}\frac{d[A]}{dt} = -\frac{1}{\beta}\frac{d[B]}{dt} = \frac{1}{\gamma}\frac{d[C]}{dt} = k[A]^{\alpha}[B]^{\beta}$$

Boolean & Quantitative Abstraction

- Data represented by molecular concentrations
- Control signals in terms of Boolean abstraction

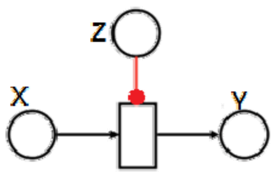
$$A_\theta = f_\theta(A) = \begin{cases} 1, & \text{if } [A] \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

Reaction Execution Precedence

- In information processing, computation must be performed in a proper order
- Data dependencies must be maintained to ensure operational correctness

Precondition

- For reaction $X + Z \rightarrow Y + Z$
- Z_θ is the *precondition* $\Rightarrow$ $X \xrightarrow{Z_\theta} Y$

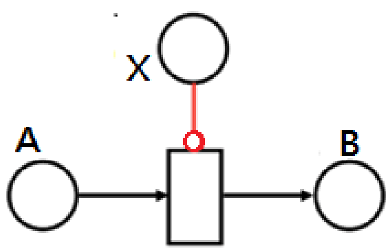


$$X \xrightarrow{A_\theta \wedge B_\theta} Y \Rightarrow X + A + B \rightarrow Y + A + B$$

$$X \xrightarrow{A_\theta \vee B_\theta} Y \Rightarrow \begin{array}{l} X + A \rightarrow Y + A \\ X + B \rightarrow Y + B \end{array}$$

Precondition

- For $X + Z \rightarrow Y + Z$,
  - At the end of the reaction, X exhausts and *Y* has the same amount as *X* before the reaction
  - We use $\neg X_\theta \wedge Y_\theta$ to denote the postcondition of the reaction
    - To represent the absence of *X*, let there be presence of some molecule, called **absence indicator**.



Absence Indicator

- Prior method [Senum and Riedel 2011]

$$\emptyset \xrightarrow{r_s} A'$$
$$A + A' \xrightarrow{r_f} A$$

Reaction rate $r_f >> r_s$

At equilibrium (when *A* is present),

$$[A'] = \frac{r_s}{r_f}[A]$$

- The amount of *A*' is sensitive to the amount of *A*
  - This “leakage” degrades the robustness
  - Can only be applied to stochastic model

Absence Indicator

- Dimerized absence indicator

$$\emptyset \xrightarrow{r_s} A'$$
$$A + A' \xrightarrow{r_f} A$$
$$2A' \underset{r_f}{\overset{r_s}{\rightleftharpoons}} A^*$$

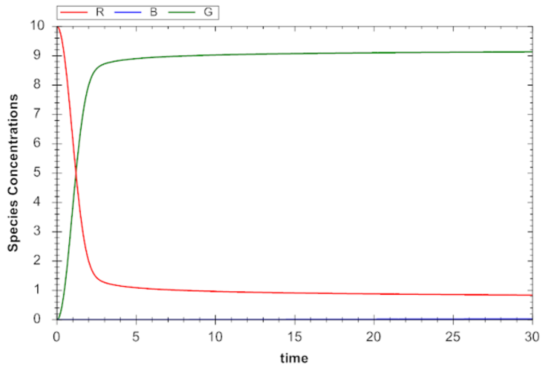
At equilibrium (when *A* is present),

$$[A^*] = \frac{r_s}{r_f}[A']^2$$

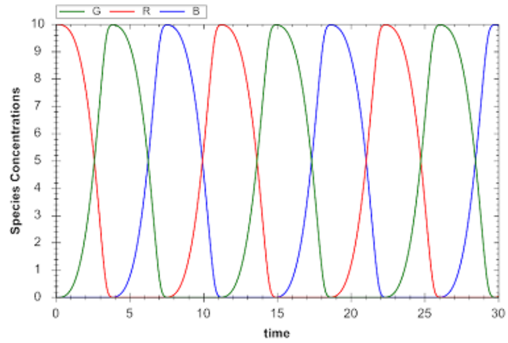
- *A** is further suppressed by the presence of *A*’
- *A** remains little even if there is a leakage of *A*

Absence Indicator

$$R \xrightarrow{\neg B_\theta} G$$
$$G \xrightarrow{\neg R_\theta} B$$
$$B \xrightarrow{\neg G_\theta} R$$



prior absence indicator



dimerized absence indicator

(by ODE simulation with SBW simulator)

Reaction Buffer

- Reaction series:

(1) $A \rightarrow B$

$B \xrightarrow{\neg A_\theta} A$

After execution for some time, A exceeds reaction threshold; precondition is violated.

(2) $A \rightarrow B$

$B \xrightarrow{\neg A_\theta} C$

$C \xrightarrow{\neg B_\theta} A$

The “buffer” reaction avoids the leakage problem

Compilation Strategy

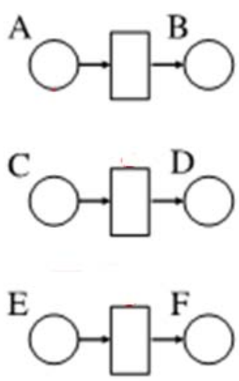
1. Identify **Linear**, **Looping**, **Branching** statements and create corresponding control flow reactions.
2. Resolve violation of *preconditions* or *postconditions*, and introduce buffer if necessary.
3. Decompose reactions for practical realization
4. Optimize

Linear Flow

- Without regulation, reactions are concurrent

Main Reactions

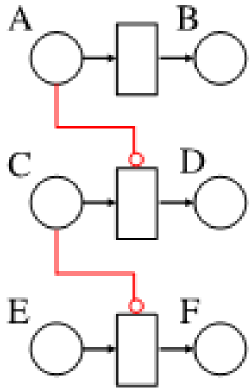
01	$A \rightarrow B$
02	$C \rightarrow D$
03	$E \rightarrow F$



Linear Flow

- When reactions are intended to be in a sequential manner, precondition can be imposed as follows:

	Main Reactions	Preconditions
01	$A \rightarrow B$	
02	$C \rightarrow D$	$\neg A_\theta$
03	$E \rightarrow F$	$\neg C_\theta$



Branching Statements

	Main Reactions	Preconditions
	$\ldots$	
01	$Q \rightarrow R$	
02	**if** $P_1(A, B)$	$\neg Q_\theta$
03	$S_1 \rightarrow T_1$	$Post(P_1)$
	$\ldots$	
04	$S_i \rightarrow T_i$	
05	$H \rightarrow I$	$\neg S_{i\theta}$
06	**else if** $P_2(A, B)$	$\neg Q_\theta$
07	$U_1 \rightarrow V_1$	$Post(P_2)$
	$\ldots$	
08	$U_j \rightarrow V_j$	
09	$H \rightarrow I$	$\neg U_{j\theta}$
10	**else**	$\neg Q_\theta$
11	$W_1 \rightarrow X_1$	$Post(\neg P_1 \wedge \neg P_2)$
	$\ldots$	
12	$W_k \rightarrow X_k$	
13	$H \rightarrow I$	$\neg W_{k\theta}$
14	$Y \rightarrow Z$	$\neg H_\theta$
	$\ldots$	

Branching Statements

	Main Reactions	Preconditions
	...	
01	$Q \rightarrow R$	
02	**if** $P_1(A, B)$	$\neg Q_\theta$
03	$S_1 \rightarrow T_1$	$Post(P_1)$
	...	
04	$S_i \rightarrow T_i$	
05	$H \rightarrow I$	$\neg S_{i\theta}$
06	**else if** $P_2(A, B)$	$\neg Q_\theta$
07	$U_1 \rightarrow V_1$	$Post(P_2)$
	...	
08	$U_j \rightarrow V_j$	
09	$H \rightarrow I$	$\neg U_{j\theta}$
10	**else**	$\neg Q_\theta$
11	$W_1 \rightarrow X_1$	$Post(\neg P_1 \wedge \neg P_2)$
	...	
12	$W_k \rightarrow X_k$	
13	$H \rightarrow I$	$\neg W_{k\theta}$
14	$Y \rightarrow Z$	$\neg H_\theta$
	...	

Same precondition to start if-else judgment

Branching Statements

	Main Reactions	Preconditions
	...	
01	$Q \rightarrow R$	
02	**if** $P_1(A, B)$	$\neg Q_\theta$
03	$S_1 \rightarrow T_1$	$Post(P_1)$
	...	
04	$S_i \rightarrow T_i$	
05	$H \rightarrow I$	$\neg S_{i\theta}$
06	**else if** $P_2(A, B)$	$\neg Q_\theta$
07	$U_1 \rightarrow V_1$	$Post(P_2)$
	...	
08	$U_j \rightarrow V_j$	
09	$H \rightarrow I$	$\neg U_{j\theta}$
10	**else**	$\neg Q_\theta$
11	$W_1 \rightarrow X_1$	$Post(\neg P_1 \wedge \neg P_2)$
	...	
12	$W_k \rightarrow X_k$	
13	$H \rightarrow I$	$\neg W_{k\theta}$
14	$Y \rightarrow Z$	$\neg H_\theta$
	...	

Design judgment carefully.
Use mutually exclusive postconditions to enter different code block.

Branching Statements

Main Reactions	Preconditions
...	
$Q \rightarrow R$	
if $P_1(A,B)$	$\neg Q_\theta$
$S_1 \rightarrow T_1$	$Post(P_1)$
...	
$S_i \rightarrow T_i$	
$H \rightarrow I$	$\neg S_{i\theta}$
else if $P_2(A,B)$	$\neg Q_\theta$
$U_1 \rightarrow V_1$	$Post(P_2)$
...	
$U_j \rightarrow V_j$	
$H \rightarrow I$	$\neg U_{j\theta}$
else	$\neg Q_\theta$
$W_1 \rightarrow X_1$	$Post(\neg P_1 \wedge \neg P_2)$
...	
$W_k \rightarrow X_k$	
$H \rightarrow I$	$\neg W_{k\theta}$
$Y \rightarrow Z$	$\neg H_\theta$
...	

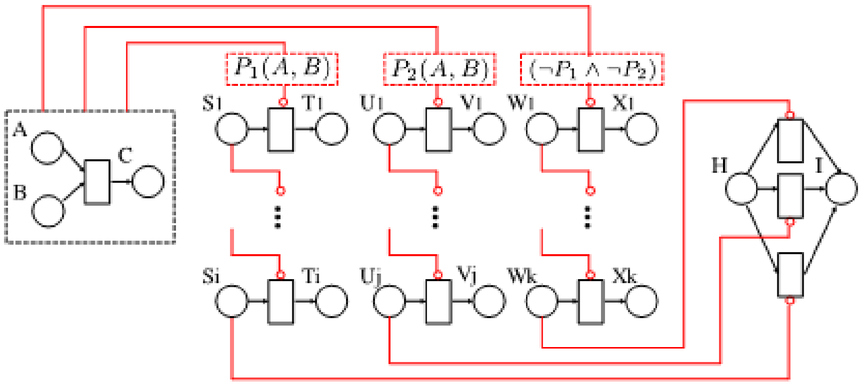
Different reaction leads to the same ending reaction

Branching Statements

Main Reactions	Preconditions
...	
$Q \rightarrow R$	
if $P_1(A,B)$	$\neg Q_\theta$
$S_1 \rightarrow T_1$	$Post(P_1)$
...	
$S_i \rightarrow T_i$	
$H \rightarrow I$	$\neg S_{i\theta}$
else if $P_2(A,B)$	$\neg Q_\theta$
$U_1 \rightarrow V_1$	$Post(P_2)$
...	
$U_j \rightarrow V_j$	
$H \rightarrow I$	$\neg U_{j\theta}$
else	$\neg Q_\theta$
$W_1 \rightarrow X_1$	$Post(\neg P_1 \wedge \neg P_2)$
...	
$W_k \rightarrow X_k$	
$H \rightarrow I$	$\neg W_{k\theta}$
$Y \rightarrow Z$	$\neg H_\theta$
...	

Same ending reaction to leave branching for each program block.

Branching Statements



Petri net visualization of branching statements:
Using $A + B \rightarrow C$ to judge (A>B)?

Looping Statement

	Main Reactions	**Preconditions**
01	$Q \rightarrow R$	
02	**while** $P(A,B)$	$\neg Q_\theta \cdot \neg F_\theta$
03	$\ldots \rightarrow F$	$Post(P)$
	$\ldots$	
04	$F \rightarrow \ldots$	
05	$X \rightarrow Y$	$Post(\neg P)$

Looping Statement

Main Reactions		Preconditions
01	$Q \to R$	
02	**while** $P(A, B)$	$\neg Q_\theta \cdot \neg F_\theta$
03	$\ldots \to F$	$Post(P)$
	$\cdots$	
04	$F \to \ldots$	
05	$X \to Y$	$Post(\neg P)$

Judgment is made when
(1)Previous reaction finished
(2)End of every single loop

Looping Statement

Main Reactions		Preconditions
01	$Q \to R$	
02	**while** $P(A, B)$	$\neg Q_\theta \cdot \neg F_\theta$
03	$\ldots \to F$	$Post(P)$
	$\cdots$	
04	$F \to \ldots$	
05	$X \to Y$	$Post(\neg P)$

Use judgment result as pre ition
to determine whether en
or not.

Loop body

Looping Statement

Main Reactions	Preconditions
$Q \rightarrow R$	
while $P(A, B)$	$\neg Q_\theta \cdot \neg F_\theta$
$\ldots \rightarrow F$	$Post(P)$
$\cdots$	
$F \rightarrow \ldots$	
$X \rightarrow Y$	$Post(\neg P)$

Executed when Looping condition not satisfied

Loop body

Case Study: Division

$Division(A,\ B)$
begin
while $A \geq B$
$A := A - B$
$Q := Q + 1$
$R := A$
end

Case Study: Division

Main Reactions	Preconditions
while $[A] \geq [B]$	
$(A + B \to D)$	$\neg G_\theta$
$C \to Q + E$	$A_\theta \wedge \neg B_\theta$
$D \to F$	$\neg C_\theta$
$E \to G$	$\neg D_\theta$
$F \to B$	$\neg E_\theta$
$G \to C$	$\neg F_\theta$
$D \to R$	$\neg A_\theta$

C of unit amount initially

Case Study: Division

Main Reactions	Preconditions
while $[A] \geq [B]$	
$(A + B \to D)$	$\neg G_\theta$
$C \to Q + E$	$A_\theta \wedge \neg B_\theta$
$D \to F$	$\neg C_\theta$
$E \to G$	$\neg D_\theta$
$F \to B$	$\neg E_\theta$
$G \to C$	$\neg F_\theta$
$D \to R$	$\neg A_\theta$

A:=A-B

Q:=Q+1

Case Study: Division

Main Reactions	Preconditions
while $[A] > [B]$	
$(A + B \rightarrow D)$	$\neg G_\theta$
$C \rightarrow Q + E$	$A_\theta \wedge \neg B_\theta$
$D \rightarrow F$	$\neg C_\theta$
$E \rightarrow G$	$\neg D_\theta$
$F \rightarrow B$	$\neg E_\theta$
$G \rightarrow C$	$\neg F_\theta$
$D \rightarrow R$	$\neg A_\theta$

$A \Leftarrow A - B$
$D \Leftarrow B$
$B \Leftarrow 0$

If $A > B$, B will exhaust and A will have amount $[A] - [B]$ remains.

Case Study: Division

Main Reactions	Preconditions
while $[A] \geq [B]$	
$(A + B \rightarrow D)$	$\neg G_\theta$
$C \rightarrow Q + E$	$A_\theta \wedge \neg B_\theta$
$D \rightarrow F$	$\neg C_\theta$
$E \rightarrow G$	$\neg D_\theta$
$F \rightarrow B$	$\neg E_\theta$
$G \rightarrow C$	$\neg F_\theta$
$D \rightarrow R$	$\neg A_\theta$

$Q \Leftarrow Q + C$

The amount of Q increment by C, (We set $C = 1$).

Case Study: Division

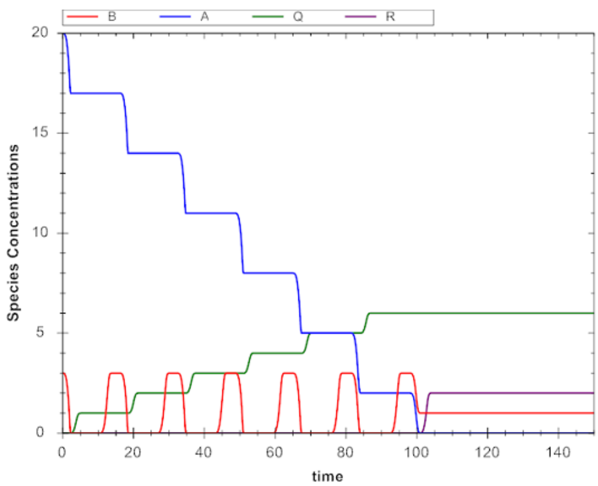
Main Reactions		Preconditions
01	**while** $[A] \geq [B]$	
02	$(A + B \rightarrow D)$	$\neg G_\theta$
03	$C \rightarrow Q + E$	$A_\theta \wedge \neg B_\theta$
04	$D \rightarrow F$	$\neg C_\theta$
05	$E \rightarrow G$	$\neg D_\theta$
06	$F \rightarrow B$	$\neg E_\theta$
07	$G \rightarrow C$	$\neg F_\theta$
08	$D \rightarrow R$	$\neg A_\theta$

$B \Leftarrow D$

B must regain value before next judgment

Case Study: Division

- Division 20÷3



(ODE simulation with SBW)

Case Study: GCD

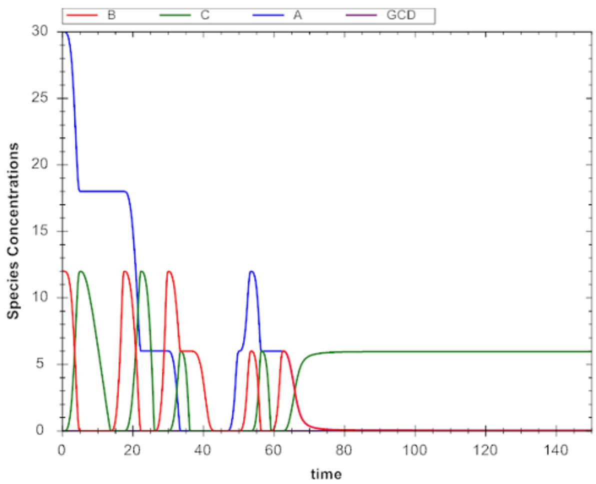
```
begin
  while A ≠ B
    if A > B
      A := A − B
    else if B > A
      swap(A, B)
  GCD := A
end
```

may cause a serious problem because [A] almost never exactly equal [B]

1000 != 1001

Case Study: GCD

- *GCD(30,12)*



(ODE simulation with SBW)

Failure due to imperfect condition A ≠ B

Case Study: GCD

```
GreatestCommonDivisor_err_toler(A, B, Z)
begin
    while |A − B| > Z
        if A > B + Z
            A := A − B
        else if B > A + Z
            swap(A, B)
    GCD := A
end
```

Z is a necessarily small constant, indicating the error tolerant range.

Case Study: GCD

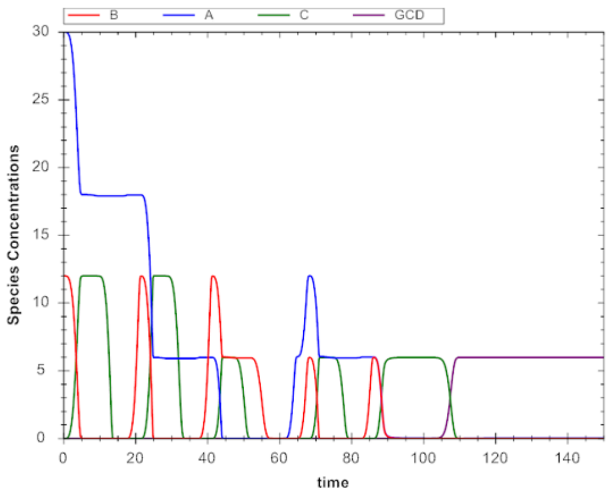
Main Reactions	Preconditions
while $\|[A]-[B]\| > [Z]$	
$(A + B \to C)$	$\neg H_\theta \wedge \neg F_\theta$
$(A + Z \to X)$	$\neg H_\theta \wedge \neg F_\theta \wedge \neg B_\theta$
$(B + Z \to Y)$	$\neg H_\theta \wedge \neg F_\theta \wedge \neg A_\theta$
if $[A] > [B] + [Z]$	
$C \to D$	$A_\theta \wedge \neg B_\theta \wedge \neg Z_\theta$
$X \to A + Z$	$\neg C_\theta \wedge \neg B_\theta$
$D \to H$	$\neg C_\theta$
$H \to B$	$\neg D_\theta$
else if $[B] > [A] + [Z]$	
$C \to E$	$\neg A_\theta \wedge B_\theta \wedge \neg Z_\theta$
$B \to G$	$\neg C_\theta \wedge \neg A_\theta$
$Y \to B + Z$	$\neg C_\theta \wedge \neg A_\theta$
$E \to F$	$\neg B_\theta$
$G \to A$	$\neg E_\theta$
$F \to A + B$	$\neg G_\theta$
$C \to GCD$	$\neg A_\theta \wedge \neg B_\theta$

$A = A - B$

$swap(A, B)$

Case Study: GCD

- *GCD*(*30*,*12*)



(ODE simulation with SBW)

Correct answer obtained by enhanced error tolerance

Future work

- Biological realization of the compilation approach
- Minimization of molecular species
- From discrete/static to continuous/dynamic computation?
  - Feedbacks
  - Robustness issues

Outline

- Compiling program control flows into biochemical reactions
- Beyond logic computation

Beyond Logic Computation

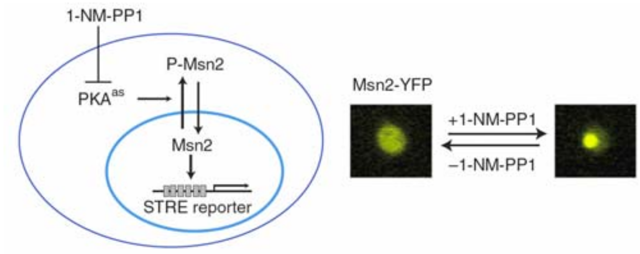
- A signal processing perspective
  - Modulators, filters, and signal processing
- A control perspective
  - Sensors, actuators (motors), and decision making

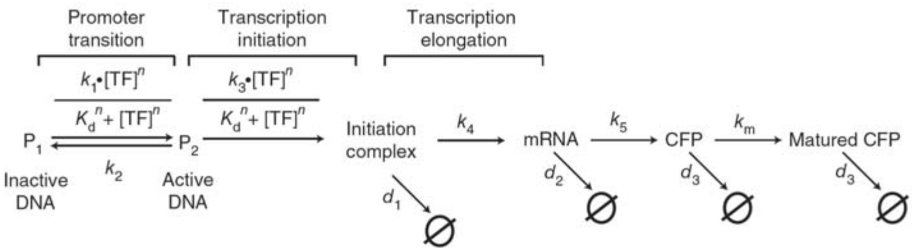
Signal Processing Perspective

Transcription factor translocation example

Nuclear translocation of *S. cerevisiae* stress response TF Msn2

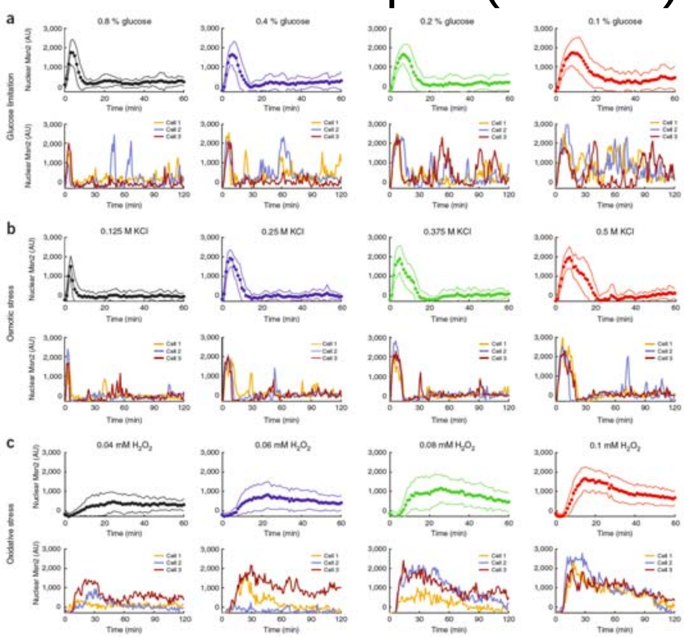
Hao and O'Shea. Nature Structural & Molecular Biology, 2012





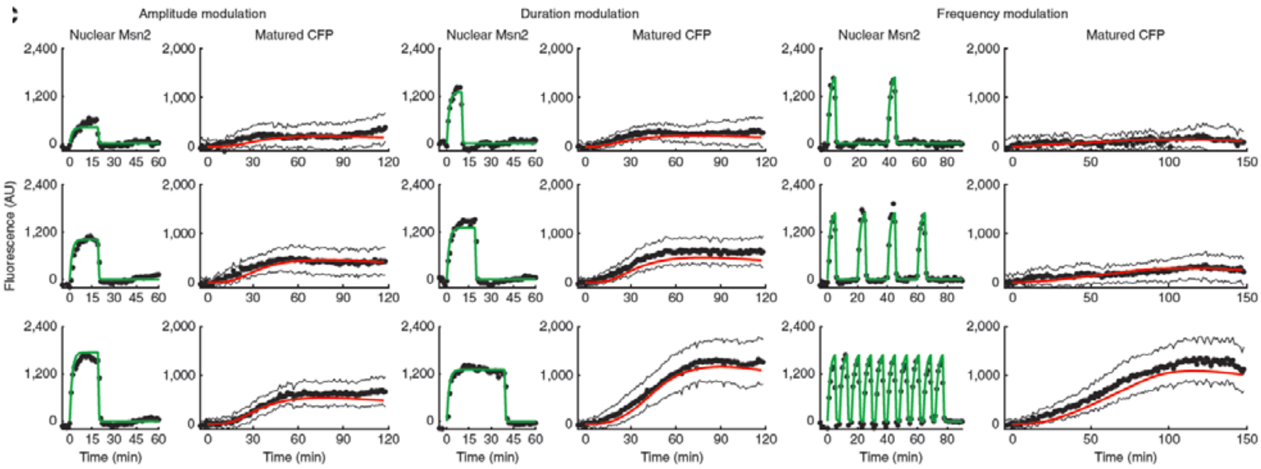
Signal Processing Perspective

TF translocation example (cont'd)



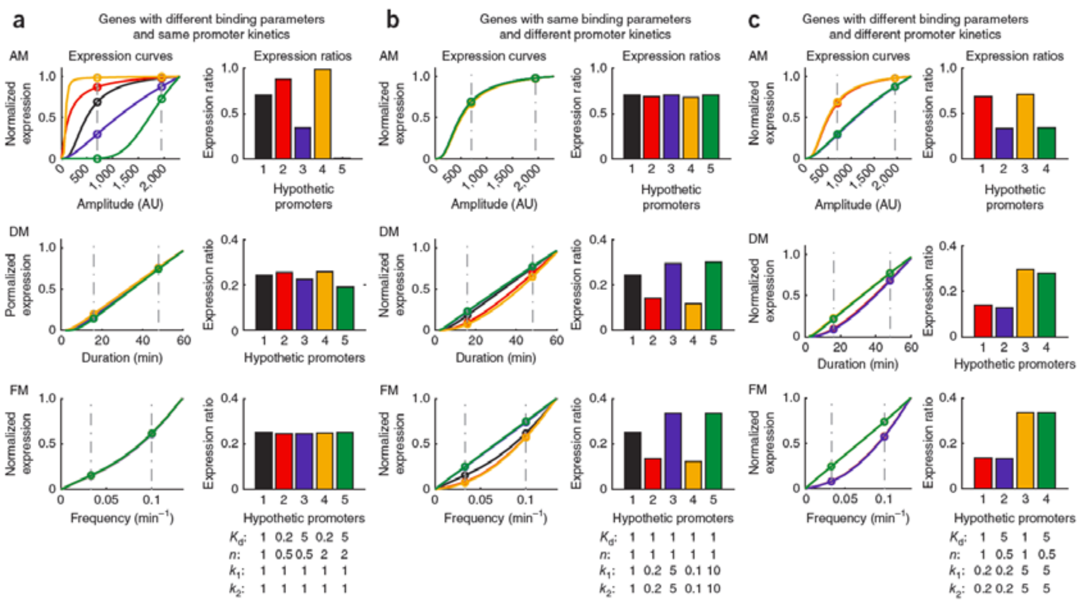
Signal Processing Perspective

- TF translocation example (cont'd)



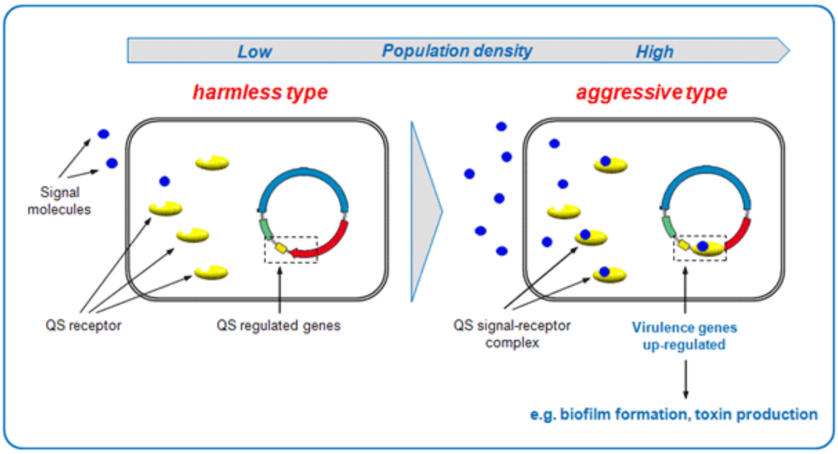
Signal Processing Perspective

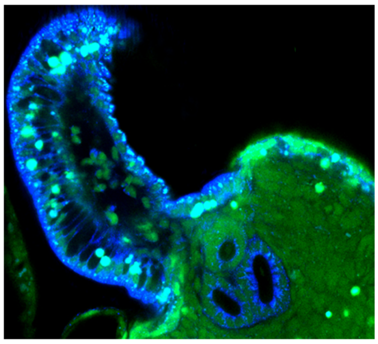
- TF translocation example (cont'd)



Control Perspective

Quorum sensing example

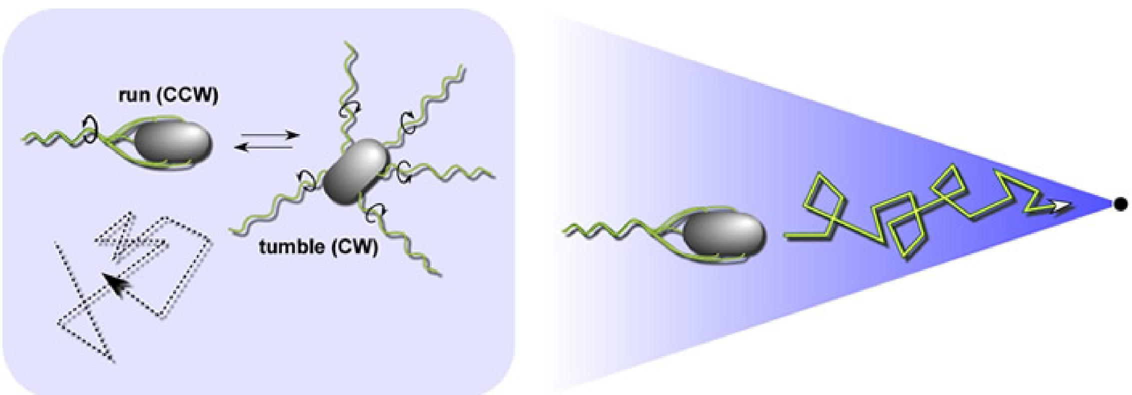




Symbiotic association between the squid Euprymna scolopes and the luminous bacterium Vibrio fischeri

Control Perspective

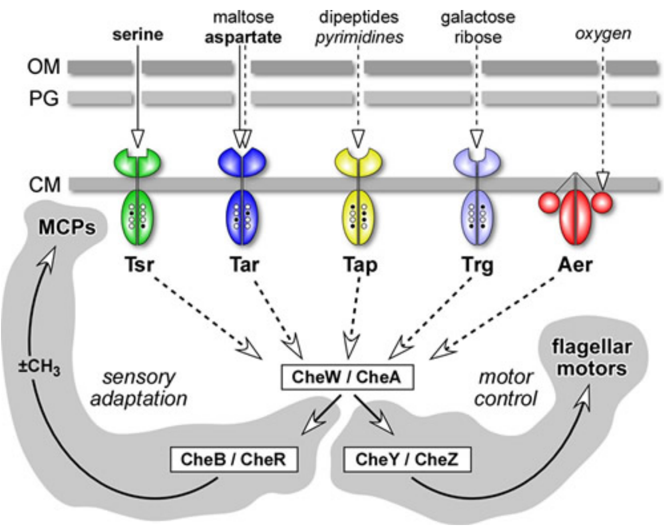
Chemotaxis example

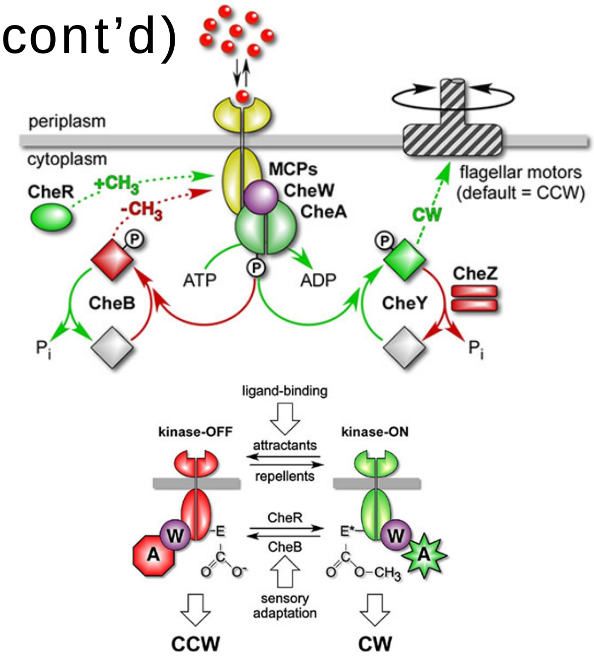


http://chemotaxis.biology.utah.edu/Parkinson_Lab/projects/ecolichemotaxis/ecolichemotaxis.html

Control Perspective

- Chemotaxis example (cont'd)

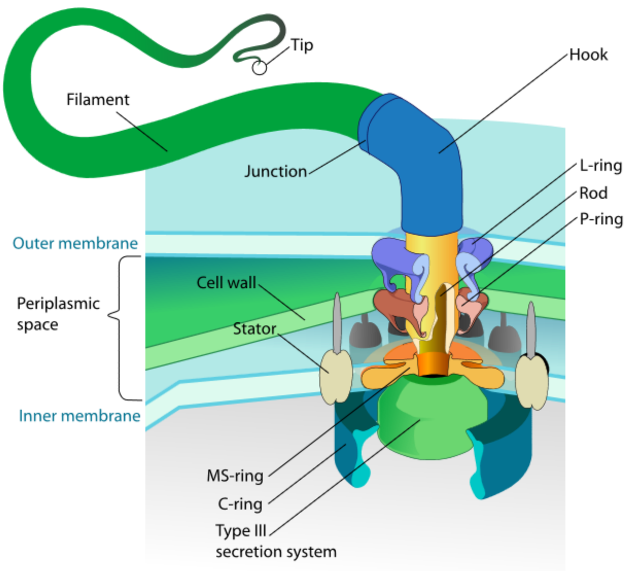


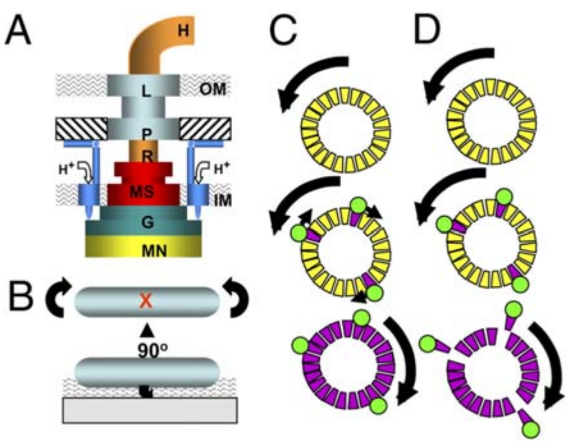


http://chemotaxis.biology.utah.edu/Parkinson_Lab/projects/ecolichemotaxis/ecolichemotaxis.html

Control Perspective

- Chemotaxis example (cont'd)





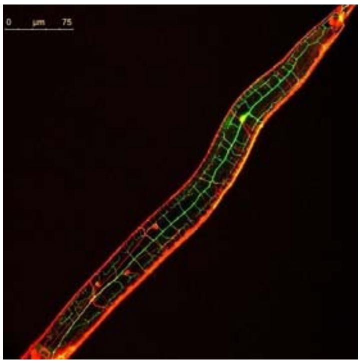
Manson M D PNAS 2010;107:11151-11152

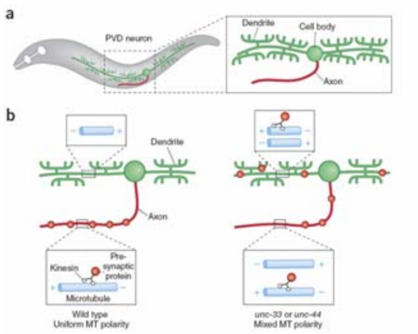
Beyond Systems Biology

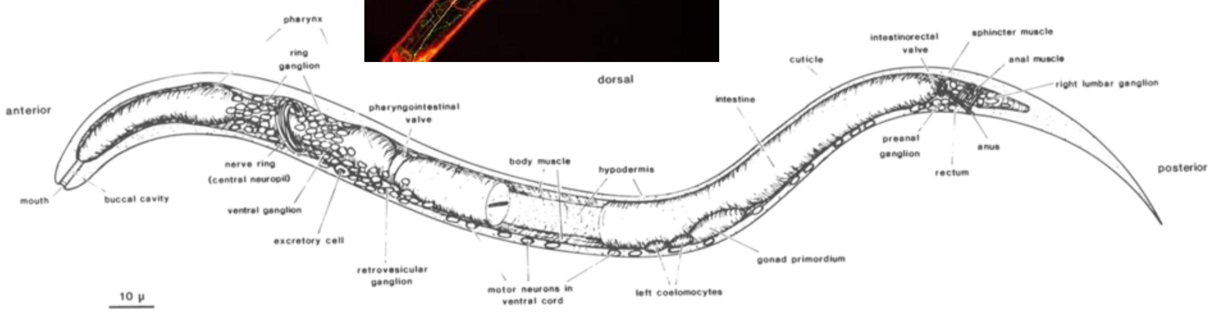
- Systems biology share strong similarities with systems neuroscience, although fundamental mechanisms are quite different
  - In biology, biochemical reactions are the fundamental mechanism
  - In neuroscience, electrical communications are the fundamental mechanism

Connection to Neuroscience

- C. elegans nervous system

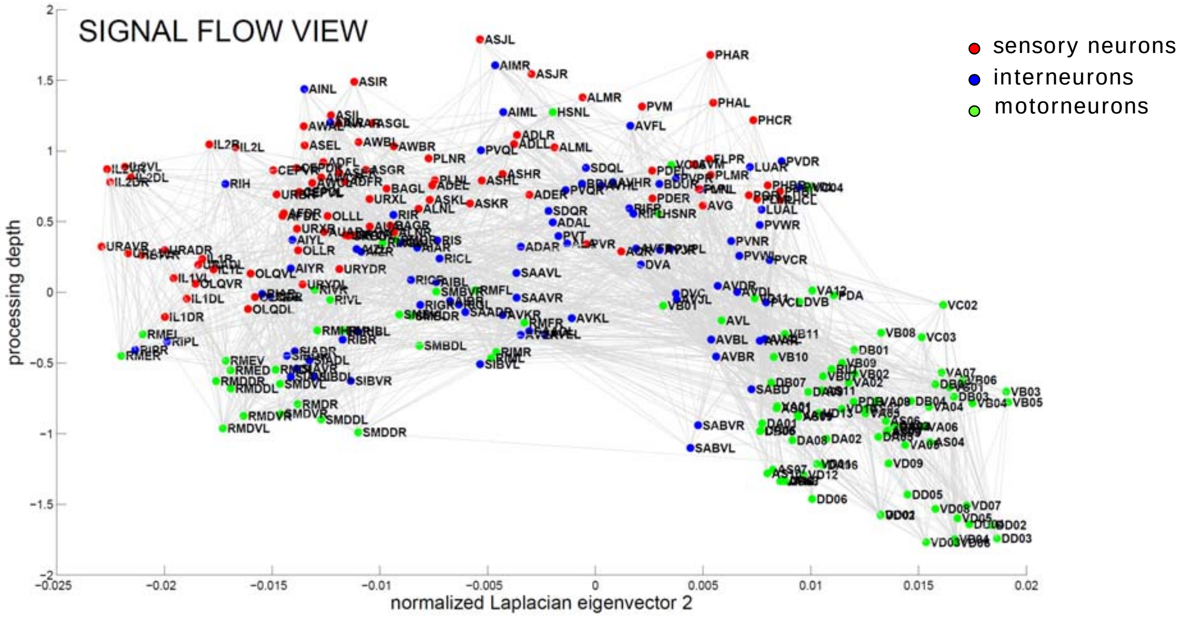






Connection to Neuroscience

- Connectome of C. elegans (302) neurons



Connection to Neuroscience

- C. elegans neuron circuitry

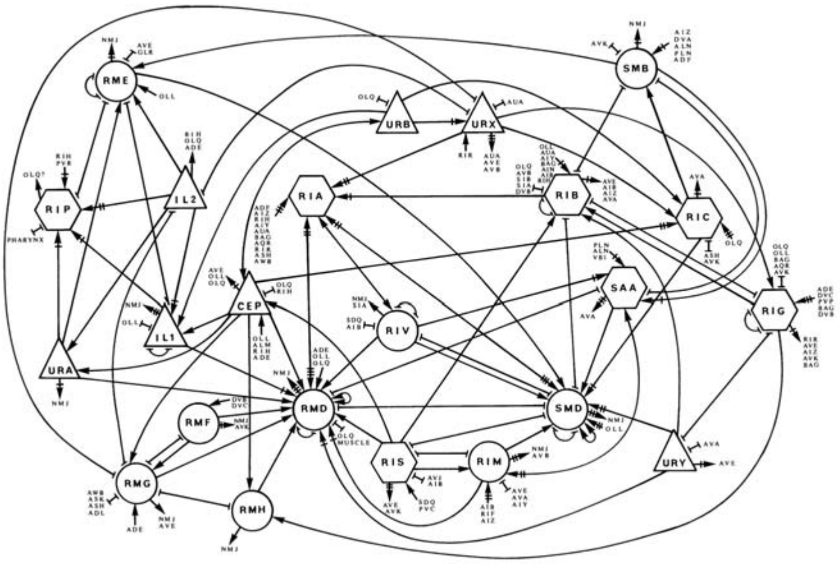


FIGURE 21. (c) Circuitry associated with the motoneurons in the nerve ring.

Summary

- Program control flows can be systematically compiled into biochemical reactions
- Discrete computation, though convenient abstraction for genetic circuits, is not a universal approach to systems and synthetic biology
- New computation models needed to decipher various open questions in systems biology and systems neuroscience