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
  - 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

\[ A + B \rightarrow C \]

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

Reaction Model

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

Example:

\[ \alpha A + \beta B \overset{k}{\rightarrow} \gamma C \]

\[ -\frac{1}{\alpha} \frac{d[A]}{dt} = -\frac{1}{\beta} \frac{d[B]}{dt} = \frac{1}{\gamma} \frac{d[C]}{dt} = k[A]^\alpha [B]^\beta \]
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
For reaction $X + Z \rightarrow Y + Z$,

- $Z_\theta$ is the *precondition* $\iff X \xrightarrow{Z_\theta} Y$
  
  \[
  Z_\theta \rightarrow X \rightarrow Y
  \]

  $X \xrightarrow{A_\theta \land B_\theta} Y \iff X + A + B \rightarrow Y + A + B$

  $X \xrightarrow{A_\theta \lor B_\theta} Y \iff X + A \rightarrow Y + A$
  $X + B \rightarrow Y + B$

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 $\neg X_\theta \land Y_\theta$ to denote the *postcondition* of the reaction
    - To represent the absence of $X$, let there be presence of some molecule, called *absence indicator*.

\[
X \rightarrow Y
\]

\[
A \rightarrow B
\]
Absence Indicator

- Prior method [Senum and Riedel 2011]
  
  Reaction rate: $r_f \gg r_s$

  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

  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

\[ R \xrightarrow{B_\theta} G \]
\[ G \xrightarrow{R_\theta} B \]
\[ B \xrightarrow{G_\theta} R \]

Prior absence indicator

Dimerized absence indicator

(by ODE simulation with SBW simulator)

Reaction Buffer

- Reaction series:

  1. \( A \rightarrow B \)
     \[ B \xrightarrow{A_\theta} A \]
     After execution for some time, \( A \) exceeds reaction threshold; precondition is violated.

  2. \( A \rightarrow B \)
     \[ B \xrightarrow{A_\theta} C \]
     \[ C \xrightarrow{B_\theta} A \]
     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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>A → B</td>
</tr>
<tr>
<td>02</td>
<td>C → D</td>
</tr>
<tr>
<td>03</td>
<td>E → F</td>
</tr>
</tbody>
</table>
Linear Flow

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

<table>
<thead>
<tr>
<th>Main Reactions</th>
<th>Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 $A \rightarrow B$</td>
<td>$\neg A_\theta$</td>
</tr>
<tr>
<td>02 $C \rightarrow D$</td>
<td>$\neg A_\theta$</td>
</tr>
<tr>
<td>03 $E \rightarrow F$</td>
<td>$\neg C_\theta$</td>
</tr>
</tbody>
</table>

Branching Statements

<table>
<thead>
<tr>
<th>Main Reactions</th>
<th>Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q \rightarrow R$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>if $P_1(A, B)$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>$S_1 \rightarrow T_1$</td>
<td>$Post(P_1)$</td>
</tr>
<tr>
<td>$S_i \rightarrow T_i$</td>
<td>$\neg S_i \theta$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>if $P_2(A, B)$</td>
<td>$Post(P_2)$</td>
</tr>
<tr>
<td>$U_1 \rightarrow V_1$</td>
<td>$\neg U_1 \theta$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>if $P_1 \land \neg P_2$</td>
<td>$Post(-P_1 \land \neg P_2)$</td>
</tr>
<tr>
<td>$W_k \rightarrow X_k$</td>
<td>$\neg W_k \theta$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg H \theta$</td>
</tr>
<tr>
<td>$Y \rightarrow Z$</td>
<td></td>
</tr>
</tbody>
</table>
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 $W_1 \rightarrow X_1$ $Post(\neg P_1 \land \neg P_2)$
... 
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.

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

<table>
<thead>
<tr>
<th>Main Reactions</th>
<th>Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q \rightarrow R$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>if $P_1(A, B)$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>$S_1 \rightarrow T_1$</td>
<td>$Post(P_1)$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg S_i \theta$</td>
</tr>
<tr>
<td>else if $P_2(A, B)$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>$U_1 \rightarrow V_1$</td>
<td>$Post(P_2)$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg U_i \theta$</td>
</tr>
<tr>
<td>else</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>$W_1 \rightarrow X_1$</td>
<td>$Post(\neg P_1 \land \neg P_2)$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg W_k \theta$</td>
</tr>
<tr>
<td>$Y \rightarrow Z$</td>
<td>$\neg H_\theta$</td>
</tr>
</tbody>
</table>

**Different reaction leads to the same ending reaction.**

<table>
<thead>
<tr>
<th>Main Reactions</th>
<th>Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q \rightarrow R$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>if $P_1(A, B)$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>$S_1 \rightarrow T_1$</td>
<td>$Post(P_1)$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg S_i \theta$</td>
</tr>
<tr>
<td>else if $P_2(A, B)$</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>$U_1 \rightarrow V_1$</td>
<td>$Post(P_2)$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg U_i \theta$</td>
</tr>
<tr>
<td>else</td>
<td>$\neg Q_\theta$</td>
</tr>
<tr>
<td>$W_1 \rightarrow X_1$</td>
<td>$Post(\neg P_1 \land \neg P_2)$</td>
</tr>
<tr>
<td>$H \rightarrow I$</td>
<td>$\neg W_k \theta$</td>
</tr>
<tr>
<td>$Y \rightarrow Z$</td>
<td>$\neg H_\theta$</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Main Reactions</th>
<th>Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 $Q \rightarrow R$</td>
<td>$\neg Q_\theta \cdot \neg F_\theta$</td>
</tr>
<tr>
<td>02 while $P(A, B)$ $\ldots \rightarrow F$</td>
<td>$Post(P)$</td>
</tr>
<tr>
<td>03 $\ldots$</td>
<td>$Post(\neg P)$</td>
</tr>
<tr>
<td>04 $F \rightarrow \ldots$</td>
<td>$Post(\neg P)$</td>
</tr>
<tr>
<td>05 $X \rightarrow Y$</td>
<td>$Post(\neg P)$</td>
</tr>
</tbody>
</table>
Looping Statement

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

Use judgment result as precondition to determine whether enter or not.
Looping Statement

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

Executed when Looping condition not satisfied

Loop body

Case Study: Division

Division($A, B$)
begin
01 while $A \geq B$
02 $A := A - B$
03 $Q := Q + 1$
04 $R := A$
end
### Case Study: Division

#### Main Reactions

<table>
<thead>
<tr>
<th>Step</th>
<th>Reactions</th>
<th>Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>while ([A] \geq [B])</td>
<td>(\neg G_\theta)</td>
</tr>
<tr>
<td>02</td>
<td>((A+B \rightarrow D))</td>
<td>(A_\theta \land \neg B_\theta)</td>
</tr>
<tr>
<td>03</td>
<td>(C \rightarrow Q+E)</td>
<td>(\neg C_\theta)</td>
</tr>
<tr>
<td>04</td>
<td>(D \rightarrow F)</td>
<td>(\neg D_\theta)</td>
</tr>
<tr>
<td>05</td>
<td>(E \rightarrow G)</td>
<td>(\neg E_\theta)</td>
</tr>
<tr>
<td>06</td>
<td>(F \rightarrow B)</td>
<td>(\neg F_\theta)</td>
</tr>
<tr>
<td>07</td>
<td>(G \rightarrow C)</td>
<td>(\neg A_\theta)</td>
</tr>
<tr>
<td>08</td>
<td>(D \rightarrow R)</td>
<td>(\neg A_\theta)</td>
</tr>
</tbody>
</table>

C of unit amount initially

#### Preconditions

- \(A_\theta\): Initially, the amount of A is 0.
- \(B_\theta\): Initially, the amount of B is 0.
- \(C_\theta\), \(D_\theta\), \(E_\theta\), \(F_\theta\), \(G_\theta\), \(A_\theta\): These are state conditions that are true at the start.

\[A := A - B\]

\[Q := Q + 1\]
Case Study: Division

If \( A > B \), B will exhaust and \( A \) will have amount \([A] - [B]\) remains.

The amount of \( Q \) increment by \( C \)

(We set \( C = 1 \)).
Case Study: Division

**Main Reactions**

<table>
<thead>
<tr>
<th>No.</th>
<th>Reaction</th>
<th>Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>while ([A] \geq [B])</td>
<td>(\neg G_\theta)</td>
</tr>
<tr>
<td>02</td>
<td>((A + B \rightarrow D))</td>
<td>(A_\theta \land \neg B_\theta)</td>
</tr>
<tr>
<td>03</td>
<td>(C \rightarrow Q + E)</td>
<td>(\neg C_\theta)</td>
</tr>
<tr>
<td>04</td>
<td>(D \rightarrow F)</td>
<td>(\neg D_\theta)</td>
</tr>
<tr>
<td>05</td>
<td>(E \rightarrow G)</td>
<td>(\neg E_\theta)</td>
</tr>
<tr>
<td>06</td>
<td>(F \rightarrow B)</td>
<td>(\neg F_\theta)</td>
</tr>
<tr>
<td>07</td>
<td>(G \rightarrow C)</td>
<td>(\neg A_\theta)</td>
</tr>
<tr>
<td>08</td>
<td>(D \rightarrow R)</td>
<td></td>
</tr>
</tbody>
</table>

\[B \leftarrow D\]

B must regain value before next judgment

---

Case Study: Division

- Division \(20 \div 3\)

![Graph](image)

(ODE simulation with SBW)
Case Study: GCD

```plaintext
begin
01 while A ≠ B
02 if A > B
03 A := A - B
04 else if B > A
05 swap(A, B)
06 GCD := A
end
```

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

1000 != 1001

Case Study: GCD

- **GCD(30,12)**

Failure due to imperfect condition A ≠ B

(ODE simulation with SBW)
Case Study: GCD

GreatestCommonDivisor_err_toler(A, B, Z)
begin
01 while |A - B| > Z
02 if A > B + Z
03 A := A - B
04 else if B > A + Z
05 swap(A, B)
06 GCD := A
end

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

Case Study: GCD

Main Reactions

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Preconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>01 while</td>
<td>![Z][A] - ![Z][B] &gt; ![Z]</td>
</tr>
<tr>
<td>02 (A + B → C)</td>
<td>![H] ∧ ![F] ∧ ![B]</td>
</tr>
<tr>
<td>03 (A + ![Z] → ![X])</td>
<td>![H] ∧ ![F] ∧ ![B]</td>
</tr>
<tr>
<td>04 (B + ![Z] → ![Y])</td>
<td>![H] ∧ ![F] ∧ ![B]</td>
</tr>
<tr>
<td>05 if ![A] &gt; ![B] + ![Z]</td>
<td>![A] ∧ ![B] ∧ ![Z]</td>
</tr>
<tr>
<td>06 C → D</td>
<td>![C] ∧ ![B] ∧ ![Z]</td>
</tr>
<tr>
<td>07 X → A + Z</td>
<td>![C] ∧ ![B]</td>
</tr>
<tr>
<td>08 D → H</td>
<td>![D]</td>
</tr>
<tr>
<td>09 H → B</td>
<td>![H]</td>
</tr>
<tr>
<td>10 else if ![B] &gt; ![A] + ![Z]</td>
<td>![A] ∧ ![B] ∧ ![Z]</td>
</tr>
<tr>
<td>11 C → E</td>
<td>![C] ∧ ![A]</td>
</tr>
<tr>
<td>12 B → G</td>
<td>![B]</td>
</tr>
<tr>
<td>13 Y → B + Z</td>
<td>![Y] ∧ ![Z]</td>
</tr>
<tr>
<td>14 E → F</td>
<td>![E]</td>
</tr>
<tr>
<td>15 G → A</td>
<td>![G]</td>
</tr>
<tr>
<td>16 F → A + B</td>
<td>![F] ∧ ![B]</td>
</tr>
<tr>
<td>17 C → GCD</td>
<td>![C] ∧ ![B]</td>
</tr>
</tbody>
</table>

A = A - B

swap(A, B)
Case Study: GCD

\[ GCD(30,12) \]

Correct answer obtained by enhanced error tolerance

(ODE simulation with SBW)

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


TF translocation example (cont’d)
TF translocation example (cont’d)
Control Perspective

Quorum sensing example

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

Chemotaxis example

http://chemotaxis.biology.utah.edu/Parkinson_Lab/projects/ecolchemotaxis/ecolchemotaxis.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

- Sensory neurons
- Interneurons
- Motorneurons

Connection to Neuroscience

- C. elegans neuron circuitry

*Figure 21. (c) Circuitry associated with the motorneurons in the nerve ring.*
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