Chemical Computation

Luke Bechtel, ECE 594
Fall 2015

Contents

Chemical Computation Overview

1) Encode Data into chemicals (Difficulty Varies)
2) Exploit Chemical Reactions to perform computation (Hard to Setup)
3) Extract Desired Data (Can be Difficult)

Dietrich Computation

- Dietrich et. al. sought to establish theoretical basis of emergent behavior in chemical computing
- Manages to minimize the difficulty of encoding and decoding data (1 & 2)

Chemical Organization Theory

The target of chemical organization theory are reaction networks.

A reaction network consists of a set of molecules M and a set of reaction rules R.

Therefore, we define a reaction network formally as a tuple $\langle M, R \rangle$ and call this tuple an **algebraic chemistry** in order to avoid conflicts with other formalizations of reaction networks.

Algebraic Chemistry & Notation

Given a set M *of molecular species and a set of reaction rules given by the relation* R : PM (M) × PM (M)*. We call the pair* ⟨M, R⟩ *an* algebraic chemistry*, where* PM (M) *denotes the set of all multisets with elements from* M*.*

A multiset differs from an ordinary set in that it can contain multiple copies of the same element. A reaction rule is similar to a rewriting operation on a multiset.

Adopting the notion from chemistry, a reaction rule is written as A → B where both A and B are multisets of molecular species. The elements of each multi set are listed with “+” symbol between them. Instead of writing {s1,s2,...,sn}, the set is written as s1 + s2 + ⋯ + sn in the context of reaction rules. We also rewrite a + a → b to 2a → b for simplicity. Note that “+” is not an operator but a separator of elements.

*

Organizations

A set of molecular species is called an organization if the following two properties are satisfied: closure and self-maintenance. A set of molecular species is closed when all reaction rules applicable to the set cannot produce a molecular species that is not in the set. This is similar to the algebraic closure of an operation in set theory.

Self-Maintenance vs. Closure

Closure: *Given an algebraic chemistry* $\langle M, R \rangle$*, a set of molecular species* $C \subseteq M$ *is closed, if for every reaction* $(A \rightarrow B) \in R$ *with* $A \in PM(C)$*, also* $B \in PM(C)$ *holds.*

Self-Maintenance: Informally, it assures that all molecules that are consumed within a self-maintaining set can also be produced by some reaction pathways within the self-maintaining set.

.

Self-Maintenance vs. Closure Example with XOR

Consider the set of Logical Reactions for the XOR Gate:

$$L = Lc = \{a+b \rightarrow c, a+B \rightarrow C, A+b \rightarrow C, A+B \rightarrow c\}$$

Given this, we can see that:

- The set {a, b} is not an organization because it is not closed. The reaction a + b → c is applicable and produces a new molecular species c that is not a member of the set {a,b}.
- The set {a,b,c} is closed but not an organization because it is not self-maintaining.

Designing A Chemical Reaction Network

Step 1: Construct a Boolean Network

A logic circuit is a composition of logic gates. As such it can be fully described by a set of boolean functions and boolean variables, forming a boolean network. Let the boolean network be defined by a set of M boolean functions and a set of N (≥ M) boolean variables:

$$\{b1,...,bM,...,bN\}$$

where $\{b_j | 1 \leq j \leq M\}$ are determined by the boolean functions (*internal variables*) and the remaining variables $\{b_j | M < j \leq N\}$ are the input variables of the boolean network. The set of boolean functions is

$$\{bi = Fi(bq(i,1),...,bq(i,ni)) \mid i = 1,...,M\}$$

where $b_{q(i,k)}$ indicates the boolean variable listed as the k-th argument of the i-th function. Since the i-th boolean function F_i takes n_i boolean variables as arguments, there are 2^{n_i} possible inputs. Thus the truth table T_i for function F_i has 2^{n_i} rows and $n_i + 1$ columns.

Step 2: Construct the Algebraic Chemistry

Given the boolean network, an algebraic chemistry $\langle M, R \rangle$ is designed as follows. For each boolean variable b_j we assign two molecular species s_{2j-1} and s_{2j} representing the value 0 and the value 1 in it, respectively. Thus the set of molecular species M contains 2N molecular species as follows:

$$M = \{s2j-1, s2j \mid j = 1,...,N\}$$

The set of reaction rules can be decomposed into two sets of reactions:

$$R = L \cup D.$$

Step 2 Cont'd (Add Logical Reactions)

Set of reactions L is derived from the logical operations of the boolean functions with $L = \bigcup_{i=1}^{M} L_i$ where L_i is a set of *logical reactions* associated with the truth table T_i of boolean function F_i. For each input case h (each row of the truth table), one reaction rule is created:

$$L_i = \{A_{i,h} \rightarrow B_{i,h} \mid h = 1,\ldots,2n_i\}.$$

The lefthand side is a set of *reactants* $A_{i,h} = \{a_{i,1,h} + \cdots + a_{i,k,h} + \cdots + a_{i,n_i,h}\}$ where $a_{i,k,h}$ is a molecular species representing the boolean variable that is taken as the k-th argument of function F_i and thus $b_{q(i,k)}$. Since two molecular species $s_{2q(i,k)-1}$ and $s_{2q(i,k)}$ are assigned to boolean variable $b_{q(i,k)}$ depending on its content, the truth table T_i is used to select from the two. If the entry $t_{ih,k}$ of the truth table is equal to 0, $b_{q(i,k)}$ must be set to 0 in the h-th input case, and thus $s_{2q(i,k)-1}$ is chosen as the reactant. Otherwise, $a_{i,k,h}$ is $s_{2q(i,k)}$:

$$a_{i,k,h} = \begin{cases} s_{2q(i,k)-1} & if\ t^i_{h,k} = 0\,, \\ s_{2q(i,k)} & if\ t^i_{h,k} = 1\,. \end{cases}$$

Similarly, the righthand side is a set of *products* $B_{i,h} = \{b_{i,h}\}$, and

$$b_{i,h} = \begin{cases} s_{2i-1} & if\ t^i_{h,n_i+1} = 0\,, \\ s_{2i} & if\ t^i_{h,n_i+1} = 1\,, \end{cases}$$

Step 2 Cont’d (Add Destructive Reactions)

The other component of set R is the set of *destructive reactions* D. Since binary states of a boolean variable b_j are coded with two molecular species s_{2j-1} and s_{2j} , the state becomes undefined when both or neither of the species are present. In order to avoid such a case, the two opposite molecular species are defined to vanish upon collision:

$$D=\{s2j-1+s2j \rightarrow \varnothing | j=1,...,N\}$$

Dietrich Algorithm

Input: Boolean network given by two sets: a set of M boolean functions $\{F_1, \ldots, F_M\}$ and a set of N boolean variables $\{\mathtt{b}_1, \ldots, \mathtt{b}_M, \ldots, \mathtt{b}_N\}$. Variables $\{\mathtt{b}_1, \ldots, \mathtt{b}_M\}$ are determined by the boolean functions (*internal variables*); the remaining variables $\{\mathtt{b}_{M+1}, \ldots, \mathtt{b}_N\}$ are input variables of the boolean network.

Output: Algebraic chemistry $\langle \mathcal{M}, \mathcal{R} \rangle$ (a set of molecular species $\mathcal{M}$ and a set of reaction rules $\mathcal{R}$) representing the boolean network without any input variable specified.[a]

Algorithm:

1. For each boolean variable $\mathtt{b}_j$:
   (a) Add two molecular species, b_j and B_j, to $\mathcal{M}$;[b]
   (b) Add one *destructive reaction* of the form $b_j + B_j \to \emptyset$ to $\mathcal{R}$;
2. For each boolean function F_i:
   (a) Create the truth table of F_i with 2^{n_i} input cases (where n_i is the arity of F_i);
   (b) For each input case, create a *logical reaction*.[c]
      i Lefthand side (*reactants*) corresponds to the input of F_i.
      ii Righthand side (*products*) consists of one molecular species representing the respective boolean output of F_i.

[a] Specifying an input variable of the boolean network is coded by an inflow reaction.

[b] As a naming convention of molecular species in this paper, the lowercase species represents value 0 in the boolean variable, and the uppercase stands for 1.

[c] For example, the XOR-function is converted into reactions as follows:

$\mathtt{b}_2$	$\mathtt{b}_3$	$\mathtt{b}_1 = F_1(\mathtt{b}_2, \mathtt{b}_3)$
0	0	0
0	1	1
1	0	1
1	1	0

$\Rightarrow$

Reactants	$\to$	Products
$b_2 + b_3$	$\to$	b_1
$b_2 + B_3$	$\to$	B_1
$B_2 + b_3$	$\to$	B_1
$B_2 + B_3$	$\to$	b_1

KEY POINTS

1. You need a full truth table for the function you want to implement.
2. For each binary variable in the truth table, you specify two separate chemicals to represent its two possible states.

Implementing an XOR

Hasse Diagram for XOR Gate

We visualize the set of all organizations by a Hasse diagram, in which organizations are arranged vertically according to their size in terms of the number of their members (e.g. Figure 1). Two organizations are connected by a line if the lower organization is contained in the organization above and there is no other organization in between.

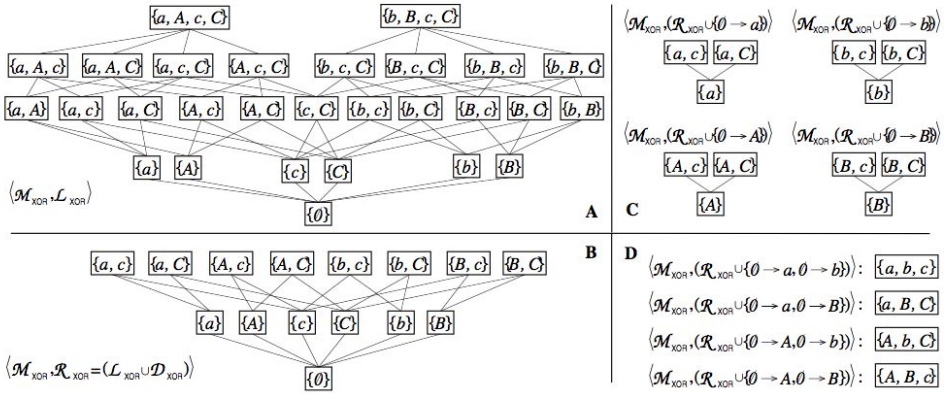

FIGURE 1
Hierarchy of organizations for the chemical reaction network implementing an XOR logic gate. (A) The network consists only of the logical reactions $\mathcal{L}_{\text{XOR}}$. (B) Destructive reactions $\mathcal{D}_{\text{XOR}}$ are added to exclude contradictions. The resulting algebraic chemistry $\langle \mathcal{M}_{\text{XOR}}, \mathcal{R}_{\text{XOR}} \rangle$ implements the XOR logic gate without any input specified. (C) One input is defined by adding one influx reaction. (D) Adding the second input. The hierarchy of organizations collapses from (A) to (D), with the desired output as the only organization left in (D).

Demo

Simulated Dietrich Computer used to calculate XOR

Observations

- Large amounts of particles (>500) on my computer are slow
- Requiring the truth table for your computation severely limits your ability to perform computations with great numbers of inputs/outputs.

Other Implemented Structures

- Flip-Flop
- Controllable Oscillator

Thank You!

Questions?