Process Algebras
&
Network Motifs

section 3

Process algebras and network motifs 3

- Introduction
  - Goals
  - Methods
  - Motivations
- Review of π–calculus
  - Syntax
  - Structural Equivalence
  - Semantics
  - Stochastics
- **Review of Kinetic Proofreading**
  - **Origins**
  - **Dynamics**
  - **Examples**
  - **Modeling in π–calculus**
- Introduction of reflective calculi
  - Syntax
  - Structural Equivalence
  - Semantics
  - A New Approach to Stochastics
  - Modeling in a reflective calculus

Algebra or logic?

Is working up to weak bisimulation sufficient?

If the algebraic notation is simply an alternative notation for the logic -- that’s fine... but then we are still missing something, according to the proposition-as-types paradigm...

Algebra or logic?

```
class Reaction {
          Reagent[] _reagents;
          Reagent[] _resultants;
          float     _basal_rate;
    …
          public Solution reduce(Solution s) {
                Solution ans = s.copy();
                for r in _reagents {
                       ans.remove( r );
                }
          for r in _resultants {
                       ans.add( r.copy() );
                }
                return( ans );
          }
  …
  }
```

Algebra or logic?

```
…
        public Solution reduce(Solution s) {
                int i = random( 10000 );
                int p = nthDigitOfPi( i );
                Solution ans = s.copy();
                int i = random( 10000 );
                int p = nthDigitOfPi( i );
                for r in _reagents {
                        int i = random( 10000 );
                        int p = nthDigitOfPi( i );
                        ans.remove( r );
                        int i = random( 10000 );
                        int p = nthDigitOfPi( i );
                }
                …
                for r in _resultants {
                        …
                        ans.add( r.copy() );
                        …
                }
                …
                return( ans );
        }
  …
```

Algebra or logic?

- The two programs are weakly bisimilar
- Is the programmer who writes the first code including the second in her mental models of the first?
- Likewise is the biologist looking at a specific network including radical variants?
    - Under what conditions is occam’s razor at work?
    - When might she be thinking of such classes of networks?

Algebra or logic?

Is working up to weak bisimulation sufficient?

No! A language of **individuals** -- or witnesses -- is still needed!

Course check

That Michaelis-menten is a scheme or a type is **obvious**:

There are no elements or compounds mentioned in the equation

$$E + S \leftrightarrow {}^{k1}{}_{k-1}\, ES \rightarrow {}^{k2}\, E + P$$

It is understood to be **instantiated**

Network motifs

What are some of the other schemes biologists have identified?

Why these motifs?

- Because they show up in physical systems?
- Because they are statistically over-represented in physical networks?
- Because they are easy to analyze with existing tools?
- Because they cohere?
  - As a set of gadgets that may be combined they define some expressive class
    - Turing complete
    - CFG
    - PDA’s
    - Regular languages

Why these motifs?

- Analog between this question and this one:
  - Why these operators in our process calculus and not another set?
- Analog to an even more fundamental question:
  - Why this model of computation and not another?
- Must the answer come from features the domain?
  - Mobile process algebras are the only **scale-invariant** model of computation
  - Are there other invariants at work selecting network combinators?
  - How are these related to biologically relevant and realizable observations?

Kinetic proofreading -- some biological processes

- Why kinetic proofreading?
  - dna replication
  - T-cell receptor signal transduction
  - dna damage detection
- Generalizes something we have already studied

$$E + S \leftrightarrow^{k1}_{k\text{-}1} ES \rightarrow^{k2} E + P$$

$$E + S \leftrightarrow^{up00}_{dn0} ES \rightarrow^{up01} ES^{*} \rightarrow^{up02} E + P$$

- Generalizes to a family of schemes

$$E + S \leftrightarrow^{up00}_{dn0} ES \rightarrow^{up01} ES^{*} \rightarrow^{up02} E + P$$

$$E + S \leftrightarrow^{up00}_{off1} ES \rightarrow^{up01} ES^{*} \rightarrow^{up02} ES^{**} \rightarrow^{up03} E + P$$

$$E + S \leftrightarrow^{up00}_{off1} ES \rightarrow^{up01} ES^{(1)} \rightarrow^{up02} \ldots ES^{(n)} \rightarrow^{up0n+1} E + P$$

Kinetic proofreading -- some biological processes

- Why kinetic proofreading?
  - Because it provides another way to address ‘why mobile process algebras…?’
  - The family of schemes may be described recursively via composition
  - Can we do this with ode’s?

Kinetic proofreading -- mass-action analysis

$$\begin{tikzcd} E + S \arrow[r, leftrightarrow, "up00", "dn0"'] & ES \arrow[r, "up01"] & ES^* \arrow[r, "up02"] \arrow[ll, bend left] & E + P \end{tikzcd}$$

$$d[E]/dt = dn0([ES]+[ES^*])+up02[ES^*]-up00[E][S]$$

$$d[S]/dt = dn0([ES]+[ES^*]) -up00[E][S]$$

$$d[ES]/dt = up00[E][S] - dn0[ES]- up01[ES^*]$$

$$d[ES^*]/dt = up01[ES] - (dn0+up02)[ES^*]$$

$$d[P]/dt = up02[ES^*]$$

But this is not what justifies the term “proofreading”

Kinetic proofreading -- the **proofreading** part

- Define *Err* as the ratio of formation of incorrect product to correct product
- In the MM case we calculate

Err_{MM}

$=$

$(up10[E](up12/up12+dn0))/up00[E](up01/up01\_dn0)$

$=$

$(up02+dn0)/(up12+dn1)$

- In the KPR case we calculate

$Err_{KPR} = (Err_{MM})^2$

- Why? What (in)equilibrium constraints must be enforced to have this work? What assumptions must be made regarding the relative magnitudes of *dn0* and *dn1*?

Kinetic proofreading -- course check

- What is the corresponding spatial logic formula?
- Can one formula capture the entire family of schemes?
- Can one set of equations?
- Let D(n) be the set of differential equations for n steps of delay, how does one capture D (n+1)?
- Let S(n) be the spatial logic formula for n steps of delay, how does one capture S(n+1)?