Compositional exploration of combinatorial models

Kris Brown, Tyler Hanks, James Fairbanks

20.07.22
Outline

Introduction
• Why represent scientific models combinatorially?

Model space exploration
• The category of diagrams as a category of model spaces
• Example limits and colimits
• Composition recipes
• Limits and colimits: implementation

Model Selection
• Best fit chemical reaction network example
Alternative: model as opaque code / math / logic

```python
***
2 \( \text{H}_2 \) + \( \text{O}_2 \) → 2 \( \text{H}_2\text{O} \). Mass-action kinetics. Compare to experimental data and plot.
***
def main():
    # experimental data
    real_data = [0.0101, 0.012, 0.023, 0.037, 0.045, 0.053, 0.061, 0.069,
                 0.076, 0.083, 0.089, 0.096, 0.102, 0.108, 0.114, 0.119, 0.125, 0.130, 0.135,
                 0.140, 0.145, 0.150, 0.154, 0.159, 0.163, 0.167, 0.171, 0.175, 0.179, 0.183,
                 0.186, 0.190, 0.193, 0.197, 0.200, 0.203, 0.206, 0.209, 0.212, 0.215, 0.218,
                 0.221, 0.224, 0.227, 0.229, 0.232, 0.234, 0.237, 0.239]
    # Initial concentrations
    H2, O2, H2O = 1.0, 2.0, 0.0
    dt = 0.01
    results = []
    for step in range(1, 50):
        print("Step ", step)
        rate = 0.5 * H2**2 * O2
        H2 -= 2*rate*dt
        O2 -= rate*dt
        H2O += rate*dt
        results.append(H2O)
    plot(results, real_data) # Figure 4 in the paper
```

1. Generate the entire code from just declaring the reaction
2. Easily alter semantics (e.g. stochastic-based simulation)
3. Construct models compositionally (operad)
4. Construct models compositionally (limits).
5. Explore alternate reaction networks to fit the data

Many tasks we’d like to do cannot be done with arbitrary code (nor mathematical expressions).
1. Functorial generation of code

- **Input**: \( I \), \( O \)
- **Variables**: \( S \), \( T \)
- **Transition**: \( \gamma \), \( \beta \)
- **States**: \( S \), \( I \), \( R \)
- **Input Table**:
<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
- **Output Table**:
<table>
<thead>
<tr>
<th>State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
- **Transition**: infection, recovery
Specifying a reaction network as a Petri Net allows for automatically generating an ODE simulation.

\[
r_\beta = k_\beta \cdot [S] \cdot [I] \\
r_\gamma = k_\gamma \cdot [I]
\]
2. Easily change semantics with different functor

Infect:

Recover:

\[
\begin{array}{c|c}
\text{State} & \text{Name} \\
1 & S \\
2 & I \\
3 & R \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\text{Input} & \text{is} & \text{it} & \text{Output} \\
1 & 1 & 1 & 1 \\
2 & 2 & 1 & 2 \\
3 & 2 & 2 & 3 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{Transition} & \text{Name} \\
1 & \text{infection} \\
2 & \text{recovery} \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{Time} & S & I \\
0 & 0 & 0 \\
50 & 500 & 500 \\
100 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c}
\text{Number} & 0 \\
0 & 0 \\
50 & 500 \\
100 & 0 \\
\end{array}
\]
3. Build models compositionally (operad)

Database queries can be built hierarchically using a wiring diagram syntax. Raw SQL queries are not composable this way.

“Find all catalysts (that are the product of two reactions) and the reactions they catalyze”

```
SELECT state2.id
FROM S AS state1, S AS state2,
T AS tran1, T AS tran2,
I AS in1, I AS in2,
O AS out1, O AS out2
WHERE in1.is = state1.id,
in1.it = tran1.id
out1.os = state2.id
out1.ot = tran1.id
...
```

```
SELECT tran1.id, state1.id
FROM S AS state1, T AS tran1,
I AS in1,
O AS out1
WHERE in1.is = state1.id,
in1.it = tran1.id
out1.os = state2.id
out1.ot = tran1.id
```
4. Build models compositionally (limits)

High-level operations on Petri nets like products do the ‘right’ thing’ for reaction networks, unlike for symbolic syntax or raw ODEs.
5. Automated Model Selection

a.)

SIRD 2-City (True Model)

People

Time

S1

I1

S2

R2

R1

I2

D1

D2

b.)

...(SIRS)₂ : 39

SIRS : 18

(SIR)₂ : 14

SIR : 21

(SIRD)₂ : 1

(SIRSD)₂ : 1

SIRD : 5 → SIRSD : 5 → ...

c.)

Model

Data

Σ Susceptible

Σ Infected

Σ Recovered

Σ Dead

SIR

SIRS

SIRD

SIRSD
Outline

Introduction
  • Why represent scientific models combinatorially?

Model space exploration
  • The category of diagrams as a category of model spaces
  • Example limits and colimits
  • Composition recipes
  • Limits and colimits: implementation

Model Selection
  • Best fit chemical reaction network example
The category of diagrams

A particular diagram in Petri

Shape

Diagram
The category of diagrams: a lax slice category

We want pushouts and pullbacks of diagrams
Model space product

“City dimension”

“Disease dimension”
Model space pushout
Model space pushout

“Glue”
Composition

ModelExploration.jl

Catlab.jl

Example exploration workflow

Death Possibility

Disease Dimension

Transport Dimension
Composition result

Catlab.jl

ModelExploration.jl
Composition result

Catlab.jl

ModelExploration.jl
Limits - implementation

\[ P = A \times B \]

\[(x, m) \quad (f, g) \quad (y, n)\]
Limits - implementation

Product

$$P = A \times B$$

$$(x, m) (f, g) (y, n)$$

Equalizer

$$Eq$$

$$A \xrightarrow{f} B$$

$$m \xrightarrow{g} n$$

Peschke and Tholen (2020)
Colimits - implementation

Coproduct

C = A + B

Peschke and Tholen (2020)
Colimits - implementation

**Coproduct**

\[ C = A + B \]

**Coequalizer**

\[ (A, X) \xrightarrow{(F, \phi_i)} (B, Y) \xrightarrow{(H, \kappa)} (C, Z) \]

\[ (A, X) \xrightarrow{(F, \kappa)} (C, \text{Lan}_{FH}X) \]

\[ (B, Y) \xrightarrow{(H, \kappa)} (C, \text{Lan}_H Y) \]

\[ (C, Z) \]

\[ (id_C, \alpha_i) \]

\[ (id_C, \gamma) \]

\[ Peschke and Tholen (2020) \]
Outline

Introduction
• Why represent scientific models combinatorially?

Model space exploration
• The category of diagrams as a category of model spaces
• Example limits and colimits
• Composition recipes
• Limits and colimits: implementation

Model Selection
• Best fit chemical reaction network example
One particular model selection strategy

\[ \text{Loss}(y, \hat{y}, \hat{p}) = \sum_{t \in T} \sum_{s \in \{S,I,R\}} (y_s(t) - \hat{y}_s(\hat{p}, t))^2 \]
Model selection results

\( \text{Param}_{A \times B} \)

\( \text{comm. monoid} \)

\( \text{Param}_A \times \text{Param}_B \)

\( \text{concat} \)

\( \text{Param}_{A+B} \)

\( \Sigma_B \)

\( \Sigma_A \)

\( \pi_A \)

\( \pi_B \)

\( \text{Obs}_A \times \text{B} \)

\( \text{Obs}_A \)

\( \text{Obs}_{A+B} \)

\( \text{Obs}_B \)

For \((\text{SIRD})_2\), we have \(S_X: 10\%\), \(S_Y: 25\%\)

For \((\text{SIRD})_2\), we have \(S_X: 30\%, D_X: 0\%\)

S: 90\%, I: 10\% and City X: 33\%, City Y: 67\%
Model selection results

- Model
- ••••• Data
- Σ Susceptible
- Σ Infected
- Σ Recovered
- Σ Dead

SIR

SIRS

SIRD

SIRSD

SIR 2-City

SIRS 2-City

SIRD 2-City

SIRSD 2-City
Model selection results

\( SIR : 21 \)  
\( SIRS : 18 \)  
\( (SIRS)_2 : 39 \)  
\( (SIR)_2 : 14 \)  
\( (SIRD)_2 : 1 \)  
\( (SIRSD)_2 : 1 \)  
\( SIRD : 5 \)  
\( SIRSD : 5 \)  
\( \ldots \)
Future work

• More interesting “primitive” model space constructors

• Wiring diagram visualizer / GUI

• Lazy state space exploration

• Alternative applications as demonstrations (Boolean functions, circuits, NN)

• Hierarchical loss functions (optimizing overall goal + subgoals, together)
Thanks!