Tools and Techniques for Qualitative Biological Models

Jason Steggles
School of Computing Science
Newcastle University

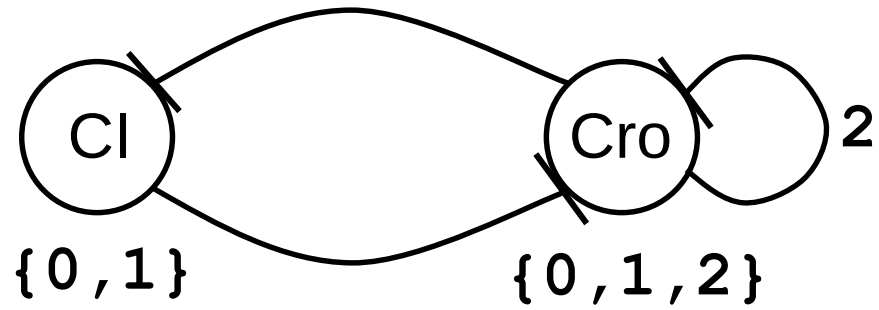
Overview

- Introduction
- Multi-Valued Networks (MVNs)
- My Work with MVNs
- Rewriting Logic Model
  - Asynchronous/Synchronous Models
- Compositional Techniques
  - Basic Idea
  - Initial Results
- Concluding Remarks

Introduction

- Qualitative state based models called **Multi-Valued Networks** (MVNs) widely used in biology.
- Used to model a range of biological systems, e.g. metabolic networks, genetic networks, …
- However, analysing and visualizing MVNs can be problematic (e.g. size of state space).
- Some good tools and techniques have been developed.
- However, more work is needed to allow them to be fully utilized.

Multi-Valued Networks (MVNs)



Next-state functions:

Cro	[CI]
0	1
1	0
2	0

CI	Cro	[Cro]
0	0	1
0	1	2
0	2	1
1	0	0
1	1	0
1	2	1

Synchronous Traces

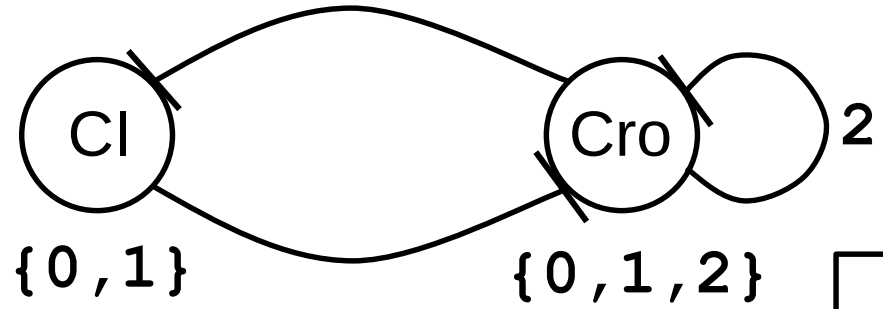
<00,11,00>
<01,02,01>
<02,01,02>
<10,10>
<11,00,11>
<12,01,02,01>

Attractor Cycles

<00,11,00>
<01,02,01>
<10,10>

(Thieffry and Thomas, 1995)

Multi-Valued Networks (MVNs)

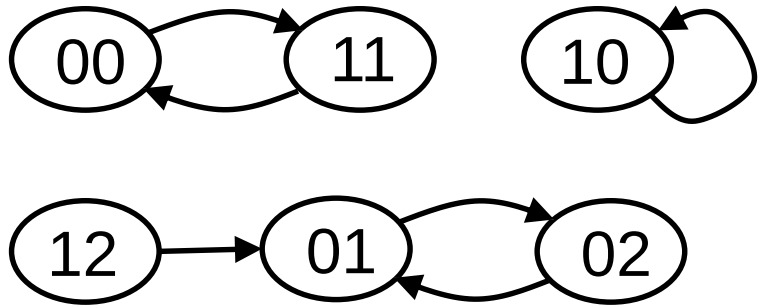


Next-state functions:

Cro	[CI]
0	1
1	0
2	0

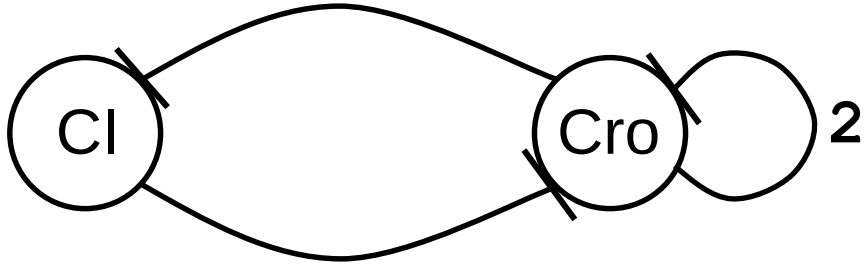
CI	Cro	[Cro]
0	0	1
0	1	2
0	2	1
1	0	0
1	1	0
1	2	1

Synchronous State Graph

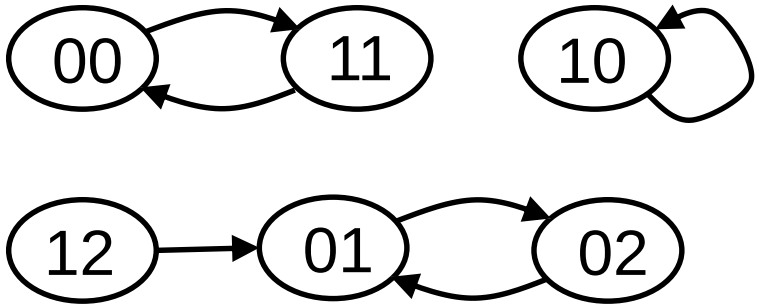


(Thieffry and Thomas, 1995)

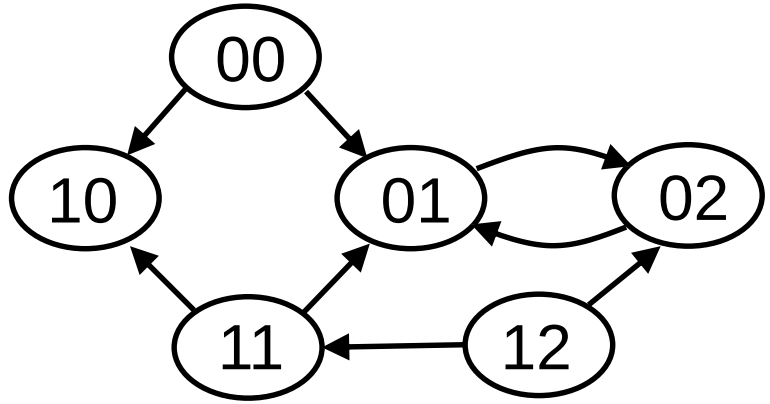
Synchronous vs Asynchronous



Sync State Graph



Async State Graph



My Work with MVNs

- **Petri Net Models of MVNs**
  - P/T Net model
  - High-Level Petri Net Model
  - Signal Transition Graphs
- **Abstraction Techniques**
  - Reducing an entities state space
- **Modelling MVNs using Rewriting Logic**
- **Compositional Techniques**

Rewriting Logic (RL)

- **Rewriting Logic** is an algebraic framework:
  - States specified by equational specifications
  - Dynamic behaviour modelled by rewrite rules
- Provides a versatile modelling framework
- Has been used for biological modelling, e.g. **Pathway Logic**
- Supported by some very good tools, e.g. **Maude**
  - Simulation and model checking tools
  - Rewrite strategies

Rewriting Logic (RL)

- States of system defined by **equational specification**:

```
subsort Sym < State .
ops a b c : -> Sym .
op _ _ : Sym Sym -> State [assoc comm] .
```

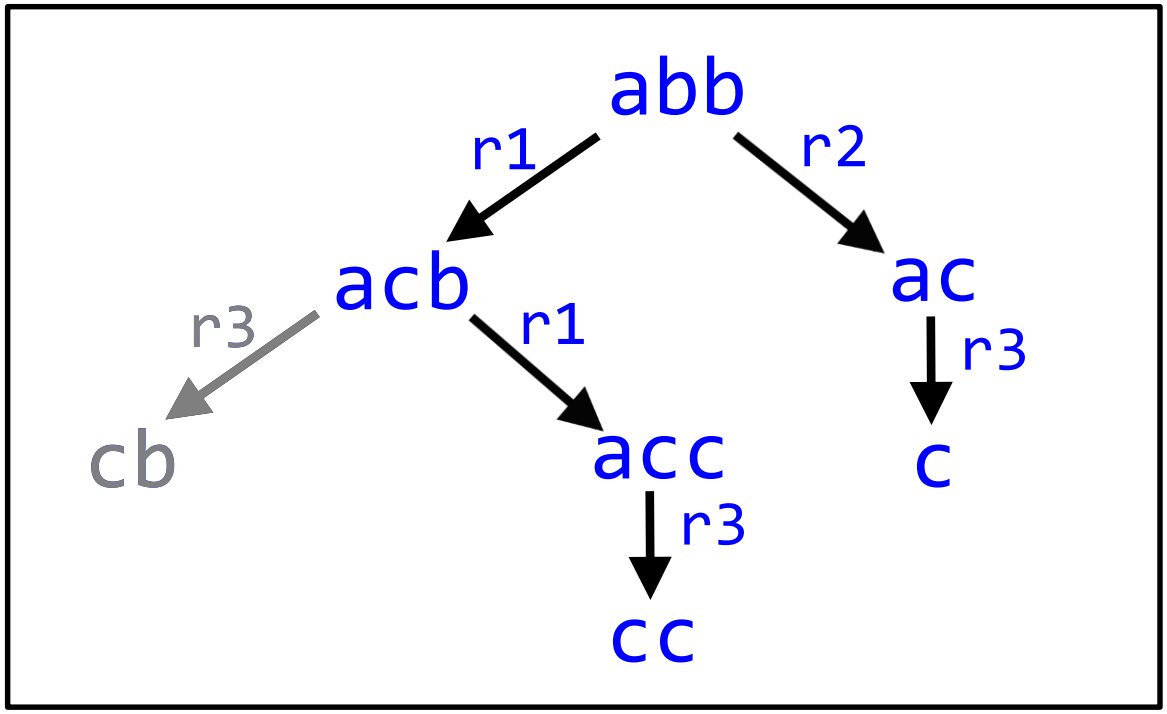
- Dynamic behaviour defined by **rewrite rules**:

```
rl [r1] : ab => ac .
rl [r2] : bb => c .
rl [r3] : ac => c .
```

- Example: $\texttt{abb} \xrightarrow{\texttt{r1}} \texttt{acb} \xrightarrow{\texttt{r1}} \texttt{acc} \xrightarrow{\texttt{r3}} \texttt{cc}$

Maude Tool

```
[r1] ab => ac
[r2] bb => c
[r3] ac => c
```



- Simulate and analyse RL models using **Maude**.
- Provides a range of model checking facilities.
- Allows **strategies** to control rewriting. `r1 ; r3`

Modelling Asynchronous MVNs

- Given an MVN MV map it to an RL model RL(MV).
- Start by defining terms to represent global states.

```
op CI : D1 -> Entity .
op Cro : D2 -> Entity .
op _ _ : Entity Entity -> GState [assoc comm].
```

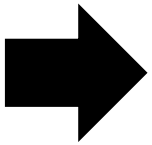
- So a global state in which CI is 1 and Cro is 2 is represented by a multi-set term:

CI(1) Cro(2)

Modelling Asynchronous MVNs

- To model **next-state functions** begin by deriving an equational specification of the MVNs behaviour.

CI	Cro	[Cro]
0	0	1
0	1	2
0	2	1
1	0	0
1	1	0
1	2	1



$$[Cro\{0\}] = CI\{1\}\ Cro\{0,1\}$$

$$[Cro\{1\}] = CI\{0\}\ Cro\{0\} + Cro\{2\}$$

$$[Cro\{2\}] = CI\{0\}\ Cro\{1\}$$

Modelling Asynchronous MVNs

- Derive rewrite rules from minimized equations.

[Cro{1}] = CI{0} Cro{0} + Cro{2}

rl CI(0) Cro(0) => CI(0) Cro(1)

rl Cro(2) => Cro(1)

Modelling Asynchronous MVNs

- Complete Set of Rewrite Rules:

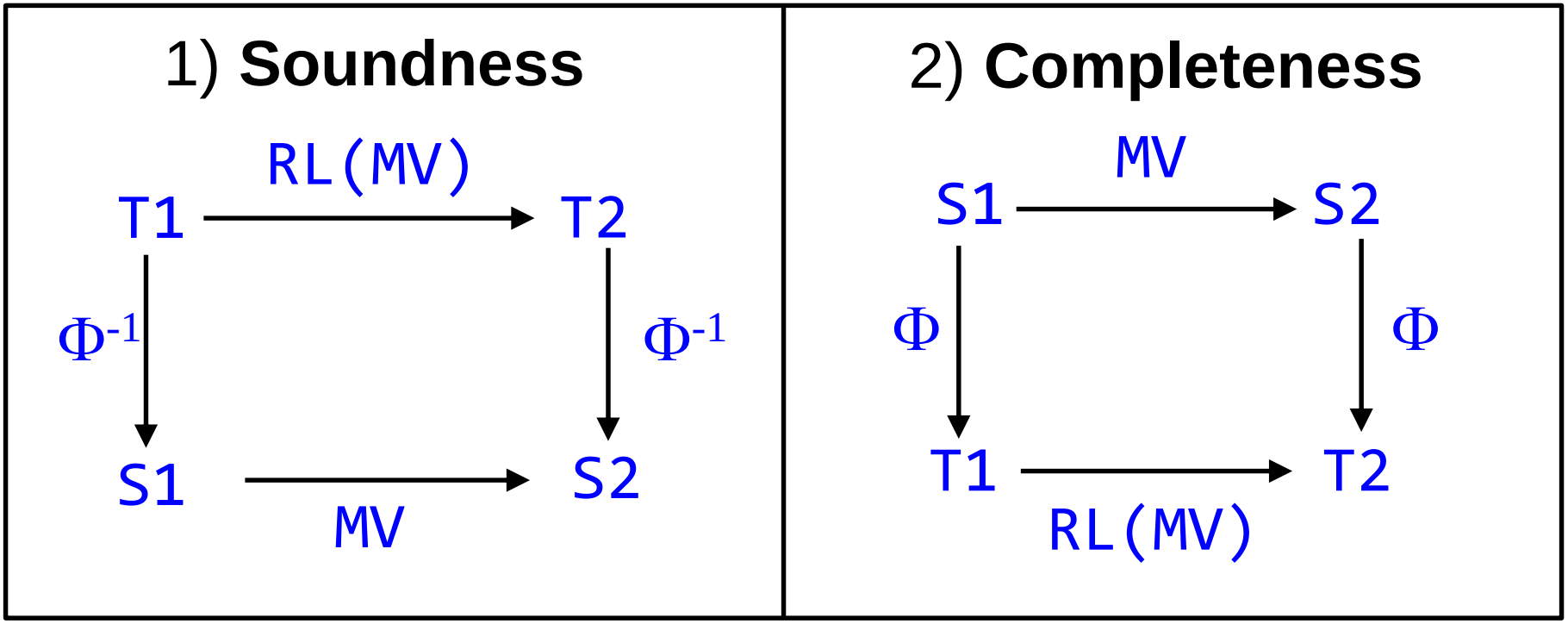
```
rl  CI(1) Cro(1) => CI(0) Cro(1) .
rl  CI(1) Cro(2) => CI(0) Cro(2) .
-----------------------------------
rl  CI(0) Cro(0) => CI(1) Cro(0) .
```

```
rl  CI(1) Cro(1) => CI(1) Cro(0) .
-----------------------------------
rl  CI(0) Cro(0) => CI(0) Cro(1) .
rl  Cro(2) => Cro(1) .
-----------------------------------
rl  CI(0) Cro(1) => CI(0) Cro(2) .
```

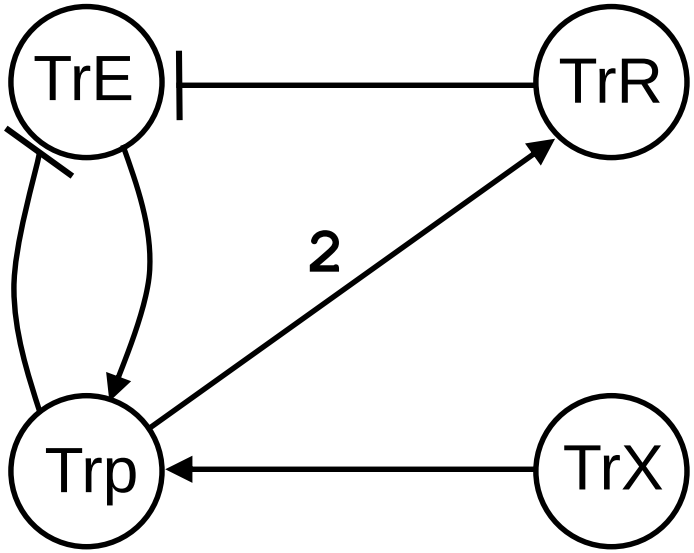
Correctness of RL Model

- Define bijective mapping from MVN to RL terms:

$$\Phi : St(MV) \rightarrow Valid(RL(MV))$$



Tryptophan Synthesis in *E. coli*



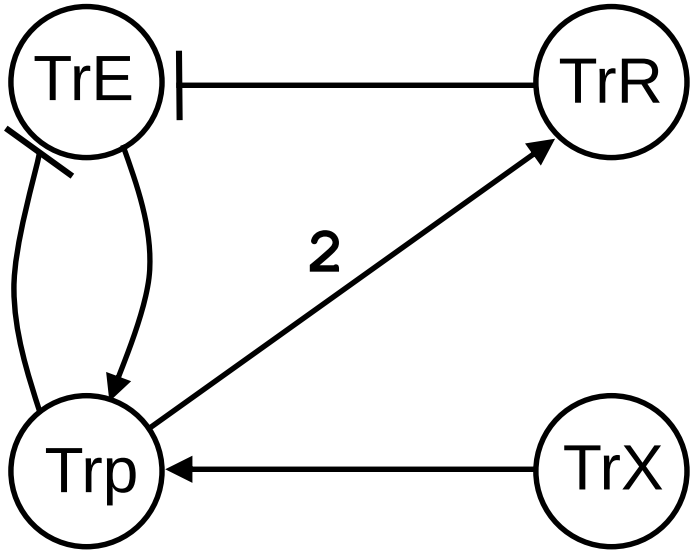
[Trp{0}] = TrE{0} TrX{0} Trp{0,1}

[Trp{1}] = TrE{0} TrX{0} Trp{2} + TrX{1} + TrX{2} Trp{0} + TrE{1} TrX{0}

[Trp{2}] = TrX{2} Trp{1,2}

(Thieffry and Thomas, 1995)

Tryptophan Synthesis in *E. coli*




```
search TrE(0) TrR(1) TrX(2) Trp(0) =>+ TrE(1) TrR(1) GS:GState .

red modelCheck(TrE(0) TrR(0) TrX(0) Trp(0), [] <> (atTrp(1) \/ atTrp(2)))
```

Modelling Synchronous MVNs

- Need to decide next state for all entities and then update states simultaneously.
- Use a **two phase state update** which we implement using a **rewriting strategy**.
- **Phase One**: alter rewrite rules to allow next state for all entities to be recorded.

```
rl  CI(0) Cro(0)  => CI(0) Cro(1)
```

```
rl  CI(0,s) Cro(0,0)  => CI(0,s) Cro(0,1)
```

Modelling Synchronous MVNs

- **Phase Two**: update the states of entities:

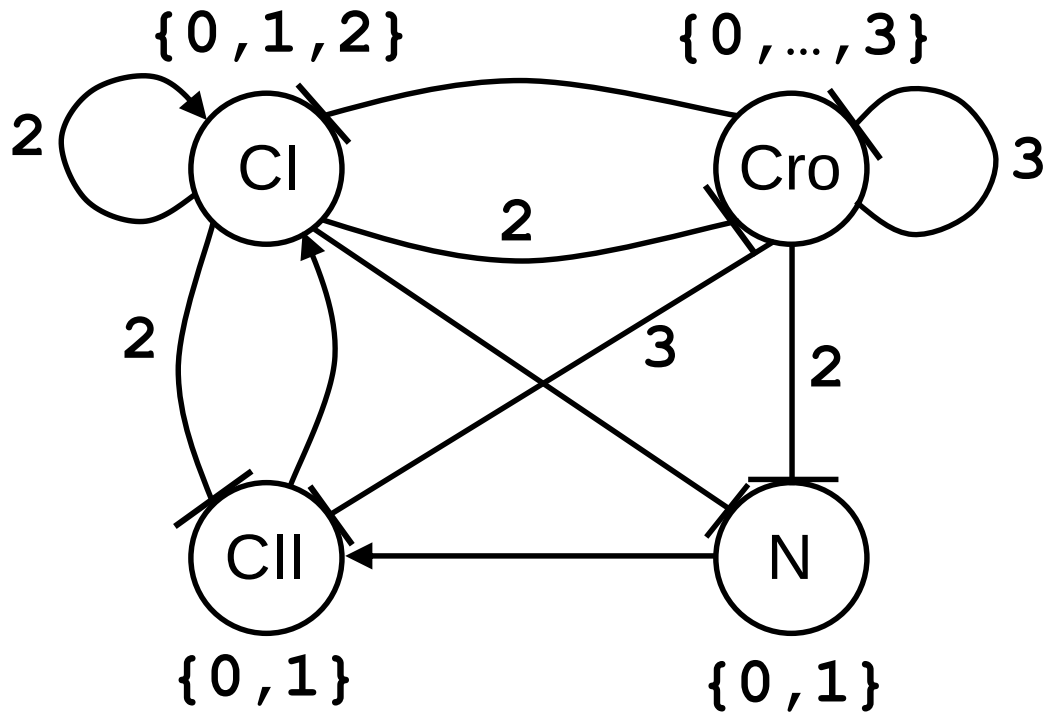
```
eq upDate(CI(s1,s2)) = CI(s2,s2) .
eq upDate(Cro(s1,s2)) = Cro(s2,s2) .
eq upDate(T1 T2) = upDate(T1) upDate(T2) .
```

- Phase one and two are combined using a **rewrite strategy** defined using Maudes meta-programming capabilities.
- Can prove new RL model of Synchronous MVNs is **sound** and **complete**.

Lysis-Lysogeny Switch in Phage Lambda

search '__['N['0.D1,'0.D1],'CII['0.D1,'0.D1],'CI['0.D2,'0.D2],'Cro['0.D3,'0.D3]]
=>+ '__['CI['1.D2,'1.D2],'CII['1.D1,'1.D1],T1:Term,T2:Term] .

red modelCheck('__['CI['1.D2,'1.D2],'CII['1.D1,'1.D1],'Cro['2.D3,'2.D3],
'N['0.D1,'0.D1]], <> [] atCI('0) -> []<> atCro('3)) .



(Thieffry and Thomas, 1995)

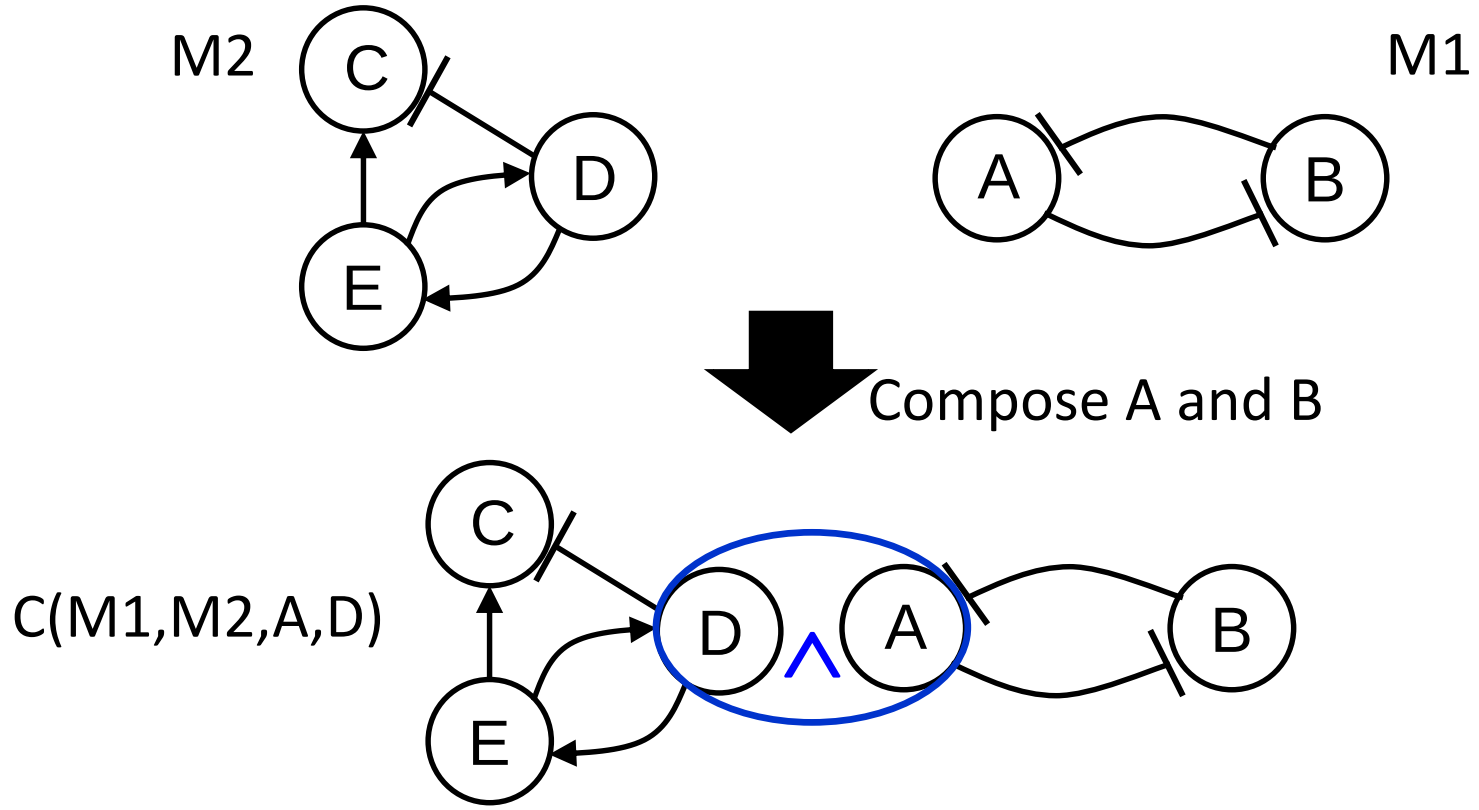
Compositional Techniques

- MVNs suffer from state space explosion problem which limits size of model that can be used.
- Would like to be able to decompose large models to make analysis tractable.
- Interested in engineering large scale biological systems from biological parts.

Solution

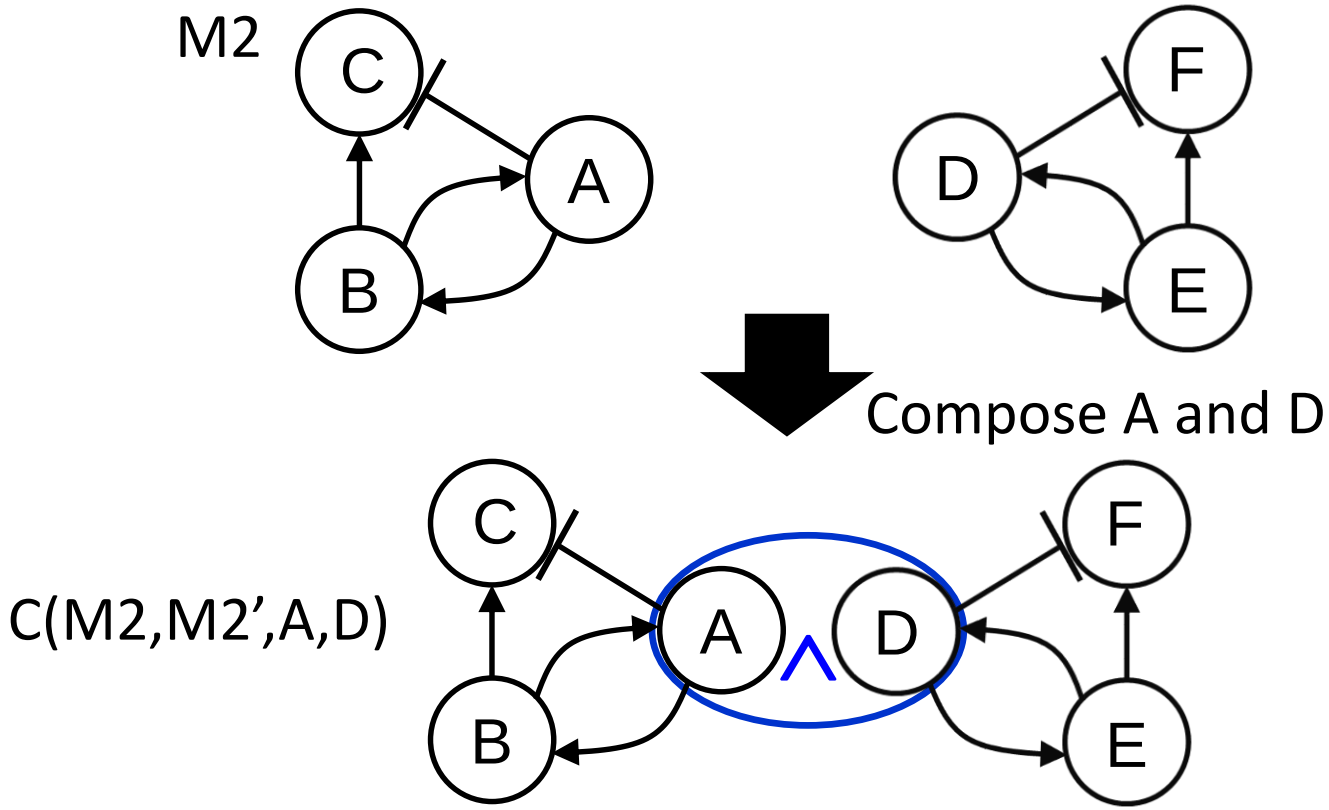
Develop a compositional framework for MVNs which allows the properties of the composed system to be derived from its subparts.

Composing Synchronous MVNS



Attractors of M1 and M2 are preserved in this example.

Composing Duplicate MVNS

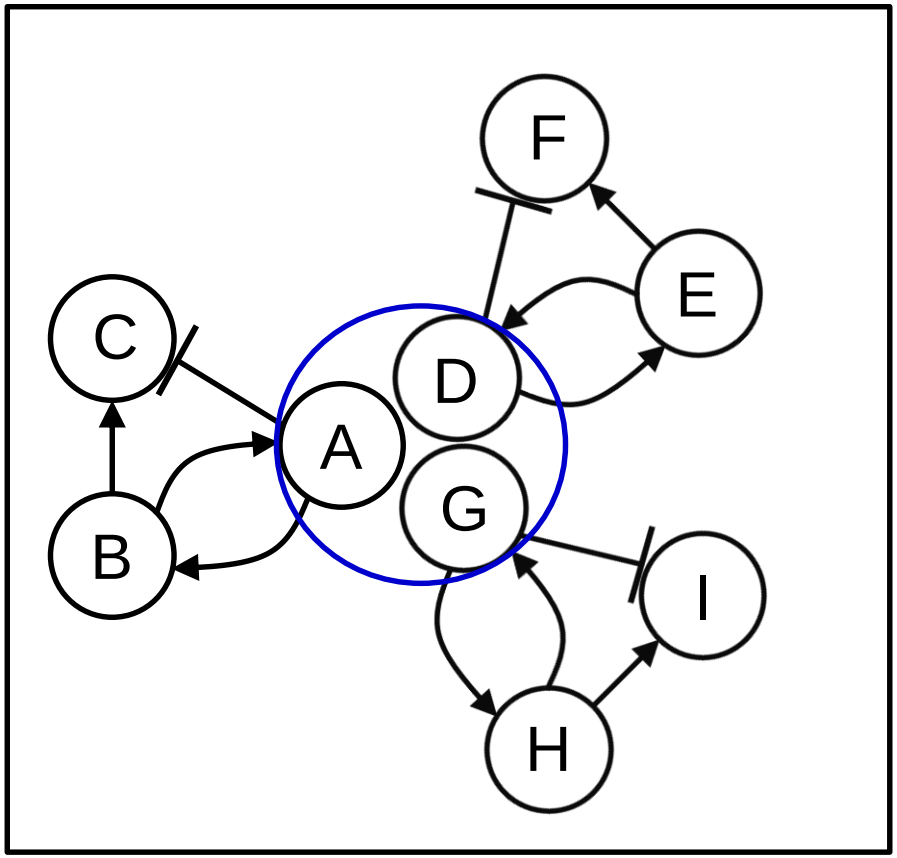


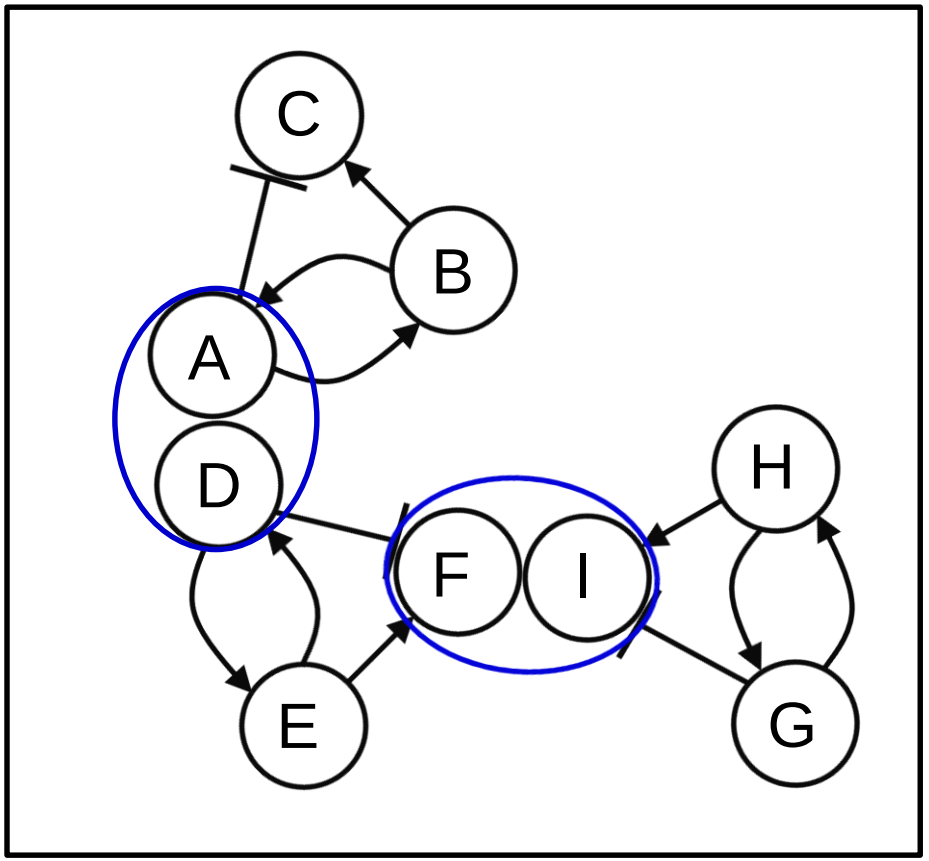
Traces of M1 and M2 are preserved in composition.

Can prove that this is always the case in this situation.

Composing Duplicate MVNS

- Traces are preserved when composing multiple duplicates on corresponding entities.





Composing Synchronous MVNS

- Have started to develop a formal framework for composing synchronous MVNs.
- Range of initial results concerning the composition of duplicate MVNs.
- Introduced the notion of two MVNs being **compatible**: their composition preserves traces.
- Working on new examples and developing results about how to check compatibility.

Concluding Remarks

- Range of work on techniques and tools for supporting MVNs.
- Recently developed an RL model of MVNs.
- Developing results for composing MVNs.
- Future work on composition:
  - Extending initial results on synchronous MVNs
  - Developing tool support based on RL and Maude.
  - Developing results for asynchronous MVNs.