SMT Solving for Vesicle Traffic Systems in Cells

A. Gupta[1] **A. Shukla**[2] M. Srivas[2] M. Thattai [3]

[1]TIFR, Mumbai

[2]CMI, Chennai

[3]NCBS, Bangalore

August 29, 2017

Biological interest: What is the form of an eukaryotic membrane traffic?

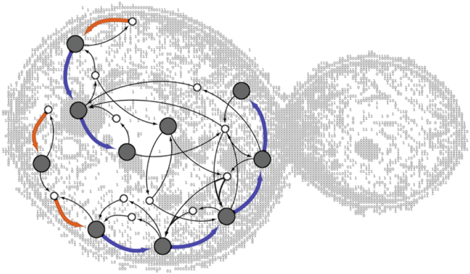
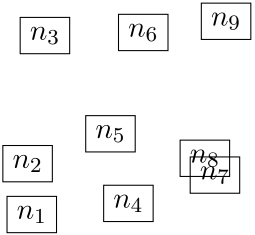
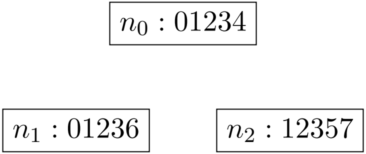


Figure from mukund's recent paper [mani16a].

Vesicular Transport in eukaryotic

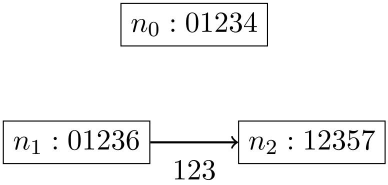


- Cells consist of compartments (nodes) 10.



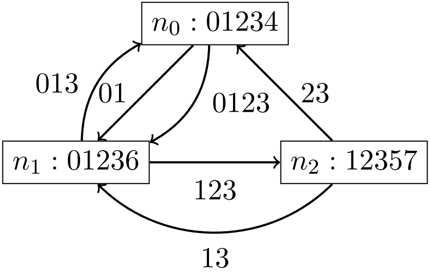
- Cells consist of compartments (nodes) 10.
- Compartments contain molecules (labels).

Vesicular Transport in eukaryotic



- Cells consist of compartments (nodes) 10.
- Compartments contain molecules (labels).
- Molecules moves around via "transfer **vesicles**" among compartments. (edges label: transferred molecules).

VTS ≡ labeled directed graph...



that follows certain biological rules.

Problem?

- Find a VTS that satisfies some **biological rules**.

[1] VTS is most complex network in cells [mani 16b]

Problem?

- Find a VTS that satisfies some **biological rules**.
- Solution?

[1] VTS is most complex network in cells [mani 16b]

Problem?

- Find a VTS that satisfies some **biological rules**.
- Solution? Easy...

[1] VTS is most complex network in cells [mani 16b]

Problem?

- Find a VTS that satisfies some **biological rules**.
- Solution? Easy...
- Encode rules as combinatorial constraints and use SAT/SMT solvers

[1] VTS is most complex network in cells [mani 16b]

Problem?

- Find a VTS that satisfies some **biological rules**.
- Solution? Easy...
- Encode rules as combinatorial constraints and use SAT/SMT solvers
- However, encoding is treacherous!! [1]

[1] VTS is most complex network in cells [mani 16b]

Problem?

- Find a VTS that satisfies some **biological rules**.
- Solution? Easy...
- Encode rules as combinatorial constraints and use SAT/SMT solvers
- However, encoding is treacherous!! [1]

Contribution: an effective encoding.

[1] VTS is most complex network in cells [mani 16b]

Find VTS that satisfied these constraints

1. Activity constraint.
2. Fusion constraint.
3. Pairing function.
4. Stability constraint.
5. Connectivity constraint.

1. "Activity constraint": a molecule may or may not be active on the edge/nodes.

[2] A VTS is k-connected if every pair of compartments remain reachable after dropping k 1 vesicles.

1. “Activity constraint”: a molecule may or may not be active on the edge/nodes.
2. “Fusion constraint”: each edge must have an active molecule that fuses with some active molecule at its target node.

[2] A VTS is k-connected if every pair of compartments remain reachable after dropping k 1 vesicles.

1. "Activity constraint": a molecule may or may not be active on the edge/nodes.
2. "Fusion constraint": each edge must have an active molecule that fuses with some active molecule at its target node.
3. "Pairing function": the fusion is dependent on molecule pairs for each edge.

[2] A VTS is k-connected if every pair of compartments remain reachable after dropping k 1 vesicles.

1. “Activity constraint”: a molecule may or may not be active on the edge/nodes.
2. “Fusion constraint”: each edge must have an active molecule that fuses with some active molecule at its target node.
3. “Pairing function”: the fusion is dependent on molecule pairs for each edge.
4. “Stability constraint”: all molecules must move around in closed cycles.

[2] A VTS is k-connected if every pair of compartments remain reachable after dropping k 1 vesicles.

1. “Activity constraint”: a molecule may or may not be active on the edge/nodes.
2. “Fusion constraint”: each edge must have an active molecule that fuses with some active molecule at its target node.
3. “Pairing function”: the fusion is dependent on molecule pairs for each edge.
4. “Stability constraint”: all molecules must move around in closed cycles.
5. “Connectivity constraint”: resulting graph is not k-connected [2].

[2] A VTS is k-connected if every pair of compartments remain reachable after dropping k 1 vesicles.

Formal VTS

A *Formal VTS* is a tuple

Definition

G = (N, M, E, L, P, f).

where

- N set of *nodes* representing compartments.
- M is a different type of *molecules* in the system.
- E is the set of edges with molecule sets as *labels*.
- L is the set of *node labels*.

Formal VTS

A *Formal VTS* is a tuple

Definition

G = (N, M, E, L, P, f).

where

- N set of *nodes* representing compartments.
- M is a different type of *molecules* in the system.
- E is the set of edges with molecule sets as *labels*.
- L is the set of *node labels*.
- P is the *pairing relation*.
- $f : M \to \wp(M) \to \mathbb{B}$ are the *activity maps*.

The search problem

Fix **k, M, N**; find a G such that following constraints are satisfied:

1. Activity constraint.
2. Fusion constraint.
3. Pairing function.
4. Stability constraint.
5. Connectivity constraint.

Results

Variant	Constraints		Graph connectivity
	Rest	Activity	
A.	F + P + S + C	N + N	No graph
B.		$\mathbb{B}$ + N	No graph
C.		N + $\mathbb{B}$	3-connected
D.		$\mathbb{B}$ + $\mathbb{B}$	2-connected
E.		N + P	No graph
F.		$\mathbb{B}$ + P	4-connected

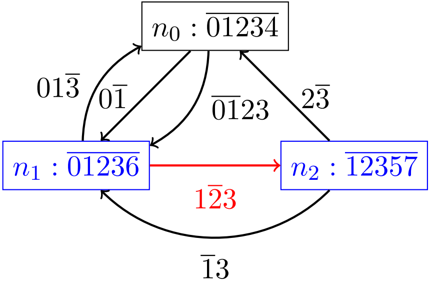
N: No regulation. Every present molecule is active.

$\mathbb{B}$: Use boolean function for regulation.

P: Use pairing matrix for regulation.

Example: Variation C

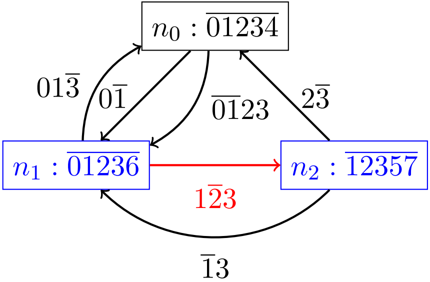
Regulation on node: **None**
Regulation on edge: **Boolean function**



- $\mathrm{N} = \{n_0, n_1, n_2\}$
- $\mathrm{M} = \{\mathrm{M0}, ..., \mathrm{M7}\}$

M1 $\rightarrow$ M6
M2 $\rightarrow$ M5
M3 $\rightarrow$ M4

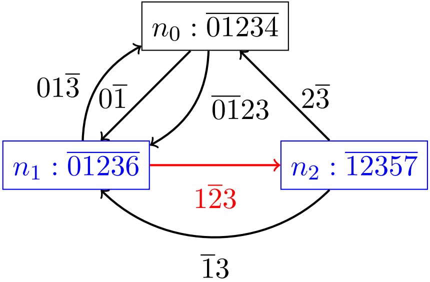
Our favorite VTS: Our favorite edge



- N = $\{n_0, n_1, n_2\}$
- M = {M0, ..., M7}
- Activity constraint.

M1 → M6
M2 → M5
M3 → M4

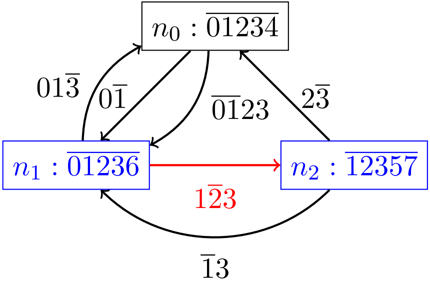
Our favorite VTS: Our favorite edge



- N = $\{n_0, n_1, n_2\}$
- M = {M0, ..., M7}
- Activity constraint.
  $1\overline{2}3$: **M2** is active.

M1 → M6
M2 → M5
M3 → M4

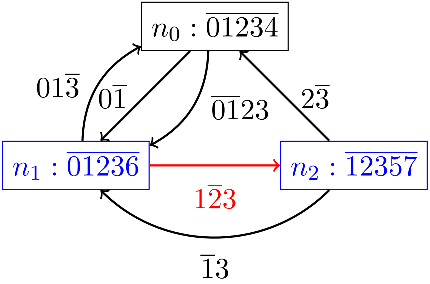
Our favorite VTS: Our favorite edge



- $N = \{n_0, n_1, n_2\}$
- M = {M0, ..., M7}
- Activity constraint.
  $1\overline{2}3$: **M2** is active.
- Fusion constraint.

M1 → M6
M2 → M5
M3 → M4

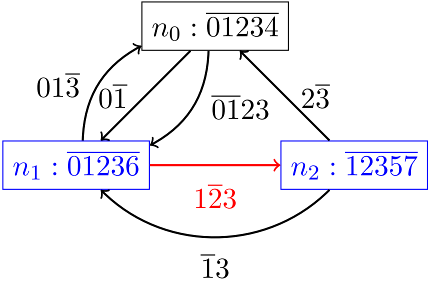
Our favorite VTS: Our favorite edge



- N = $\{n_0, n_1, n_2\}$
- M = {M0, ..., M7}
- Activity constraint.
  $1\bar{2}3$: **M2** is active.
- Fusion constraint.
  M2, M5 are active.

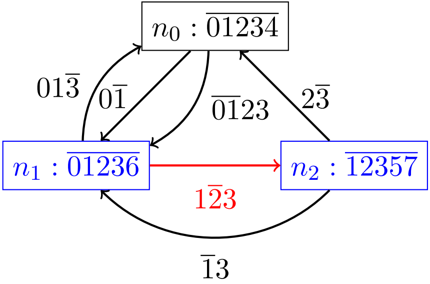
M1 $\rightarrow$ M6
M2 $\rightarrow$ M5
M3 $\rightarrow$ M4

Our favorite VTS: Our favorite edge



M1 → M6
M2 → M5
M3 → M4

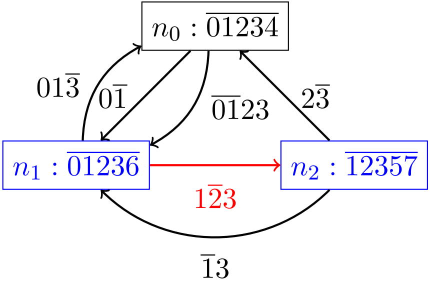
- N = $\{n_0, n_1, n_2\}$
- M = {M0, ..., M7}
- Activity constraint. $1\overline{2}3$: **M2** is active.
- Fusion constraint. **M2, M5** are active.
- Pairing function.



M1 → M6
M2 → M5
M3 → M4

- $N = \{n_0, n_1, n_2\}$
- M = {M0, ..., M7}
- Activity constraint. $1\overline{2}3$: **M2** is active.
- Fusion constraint. **M2, M5** are active.
- Pairing function. **M2, M5** form a pair.

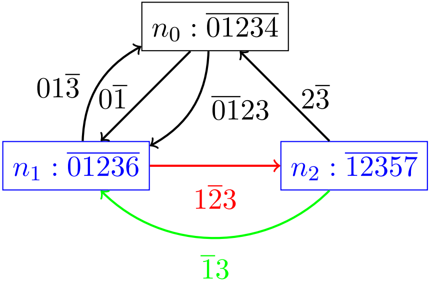
Our favorite VTS: Our favorite edge



M1 → M6
M2 → M5
M3 → M4

- N = $\{n_0, n_1, n_2\}$
- M = {M0, ..., M7}
- Activity constraint.
  $1\bar{2}3$: **M2** is active.
- Fusion constraint.
  M2, M5 are active.
- Pairing function.
  M2, M5 form a pair.
- Stability constraint.

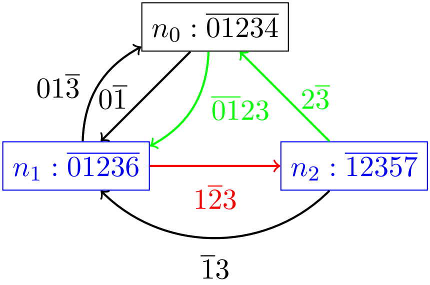
Example: Our favorite edge, Stability condition



- Stability constraint. **M1**, **M3** come back direct edge.

M1 → M6
M2 → M5
M3 → M4

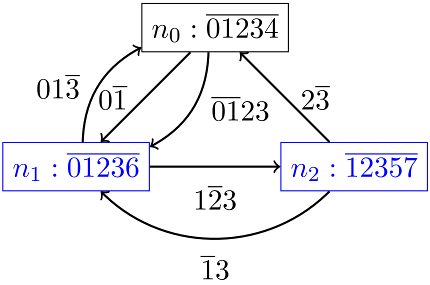
Example: Our favorite edge, SS condition



- Stability constraint. M1, M3 come back direct edge. **M2** comes back using node n_0.

M1 $\rightarrow$ M6
M2 $\rightarrow$ M5
M3 $\rightarrow$ M4

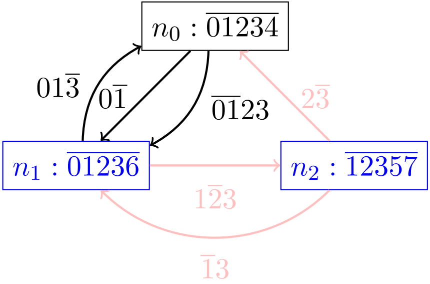
Example: Our favorite edge, CC



M1 → M6
M2 → M5
M3 → M4

- Stability constraint.
  M1, M3 come back direct edge.
  M2 comes back using node n_0.
- Connectivity constraint.

Example: Our favorite edge, CC



M1 → M6
M2 → M5
M3 → M4

- Stability constraint. M1, M3 come back direct edge. M2 comes back using node n_0.
- Connectivity constraint. Drop edge $d_{2,1}, d_{1,2}, d_{2,0}$.

1. Activity constraint.
2. Fusion constraint.
3. Pairing function.
4. Stability constraint.
5. Connectivity constraint.

Constraint for stability condition:

"*Every outgoing molecule come back in a cycle*"

$\forall$ leaving m $\in$ M... $\exists$ cycle [..]

Stability encoding

Encode stability using the reachability variables.

$$\bigwedge_{i,j,m} (e_{i,j,m} \implies$$

- Edge contains m

Stability encoding

Encode stability using the reachability variables.

$$\bigwedge_{i,j,m} (e_{i,j,m} \implies r_{j,i,m,\mu})$$

- Edge contains m
- A path btw i,j len $\leq \mu$ contains m

Stability encoding

Encode stability using the reachability variables.

$$\bigwedge_{i,j,m} (e_{i,j,m} \implies r_{j,i,m,\mu})$$

- Edge contains m
- A path btw i,j len $\leq \mu$ contains m
- μ : N − 2.

Steady state encoding

Use reachability to encode the stability condition in VTSs.

$$\bigwedge_{i,j,m,p} r_{i,j,m,p} \implies$$

- Recursively define reachability

Steady state encoding

Use reachability to encode the stability condition in VTSs.

$$\bigwedge_{i,j,m,p} r_{i,j,m,p} \implies e_{i,j,m} \vee \bigvee_{i \neq i'} (\, e_{i,i',m} \wedge$$

- Recursively define reachability

Steady state encoding

Use reachability to encode the stability condition in VTSs.

$$\bigwedge_{i,j,m,p} r_{i,j,m,p} \implies e_{i,j,m} \vee \bigvee_{i \neq i'} (e_{i,i',m} \wedge r_{i',j,m,p-1})$$

- Recursively define reachability

1. Activity constraint.
2. Fusion constraint.
3. Pairing function.
4. Stability constraint.
5. Connectivity constraint.

The following constraints encode that only existing edges can be dropped and exactly $k-1$ edges are dropped.

$$\begin{aligned} \bigwedge_{i,j} d_{i,j} \implies e_{i,j} \\ \sum_{i,j} d_{i,j} = k-1 \end{aligned} \tag{1}$$

but, in my opinion.

$$\bigwedge_{i,j} [(e_{i,j} \wedge \neg d_{i,j}) \vee (\bigvee_{i' \neq i} r'_{i',j} \wedge (e_{i,i'} \wedge \neg d_{i,i'})] \implies r'_{i,j} \tag{2}$$

$$\bigvee_{i,j} \neg(r'_{i,j} \vee r'_{j,i}) \tag{3}$$

Previous work

Old-e

- **CBMC:** C bounded model checker.
- Encode stability condition using *non-determinism* and *enumeration*.
- Not very optimal.

Run-times for searching for models (in secs)

Size	Variant A		Variant C		Variant D		Variant F	
	2-connected		3-connected		2-connected		4-connected	
	MAA	Old-e	MAA	Old-e	MAA	Old-e	MAA	Old-e
2	!0.085	!2.43	0.15	2.12	!0.13	!1.89	0.35	5.12
3	!0.54	!8.04	0.95	7.65	0.62	7.66	1.36	23.94
4	!2.57	!297.93	2.33	22.74	2.85	48.35	4.81	123.34
5	!7.7	!3053.8	7.60	500.03	10.27	890.84	33.36	2482.71
6	!22.98	M/O	19.52	M/O	30.81	M/O	147.52	M/O
7	!57.07	M/O	81.89	M/O	82.94	M/O	522.26	M/O
8	!164.14	M/O	630.85	M/O	303.19	M/O	2142.76	M/O
9	!307.67	M/O	2203.45	M/O	971.01	M/O	4243.34	M/O
10	!558.34	M/O	7681.93	M/O	2274.30	M/O	7786.82	M/O

Conclusion

1. Novel encodings of reachability and 3-4 connectivity.
2. Direct encoding into the SMT solver.
3. A user friendly and scalable tool based on well known SMT solver Z3.

Thank You !