Robustness Analysis of Networked Systems

Roopsha Samanta

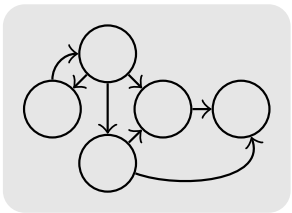
The University of Texas at Austin

Joint work with Jyotirmoy V. Deshmukh and Swarat Chaudhuri

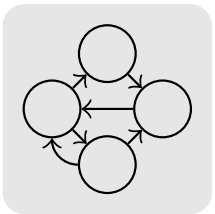
January 21, 2013

Problem Overview

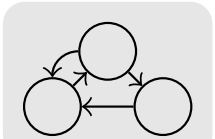
P_1



P_3

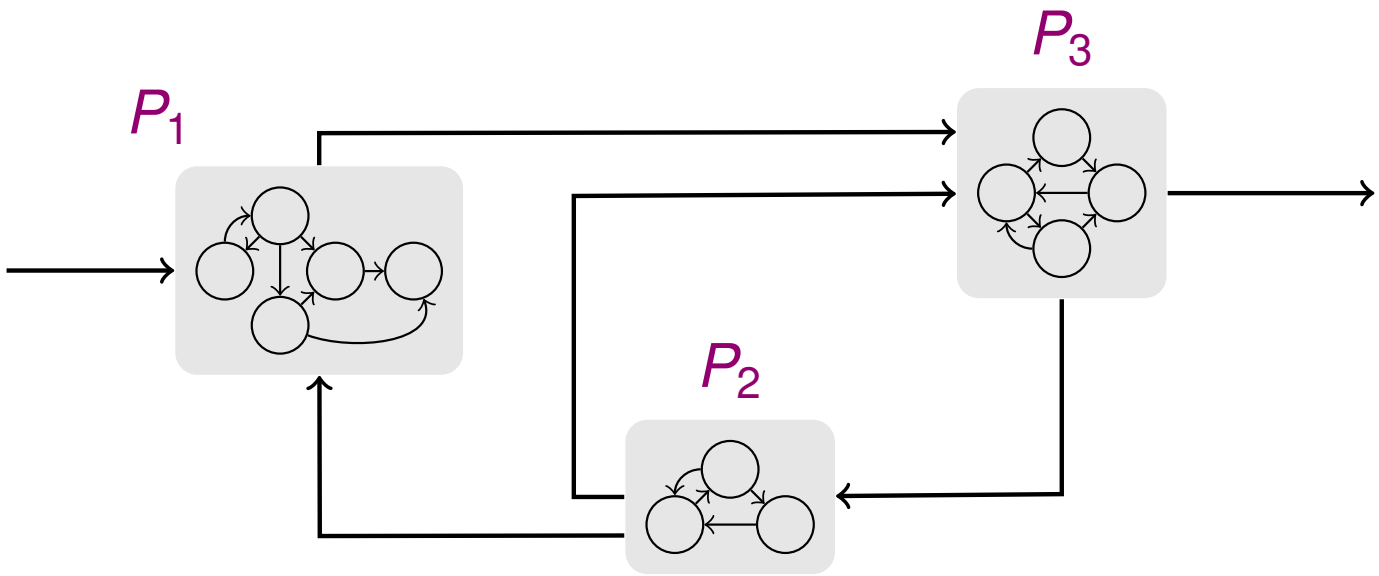


P_2



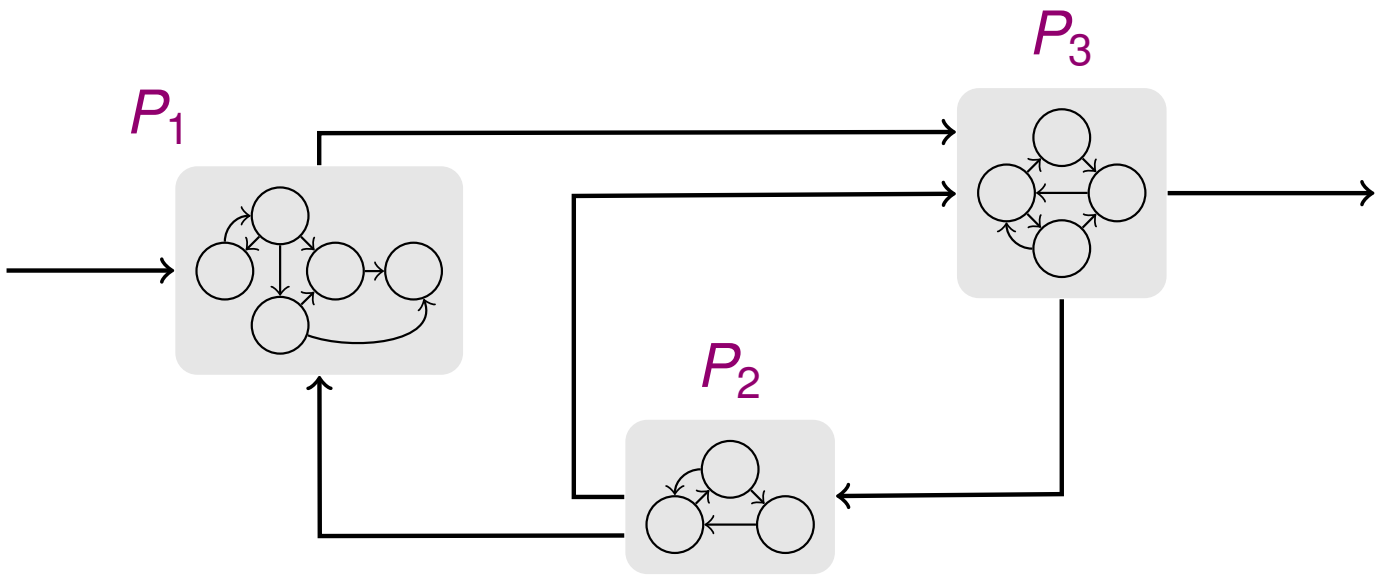
Programs rarely execute in isolation

Problem Overview



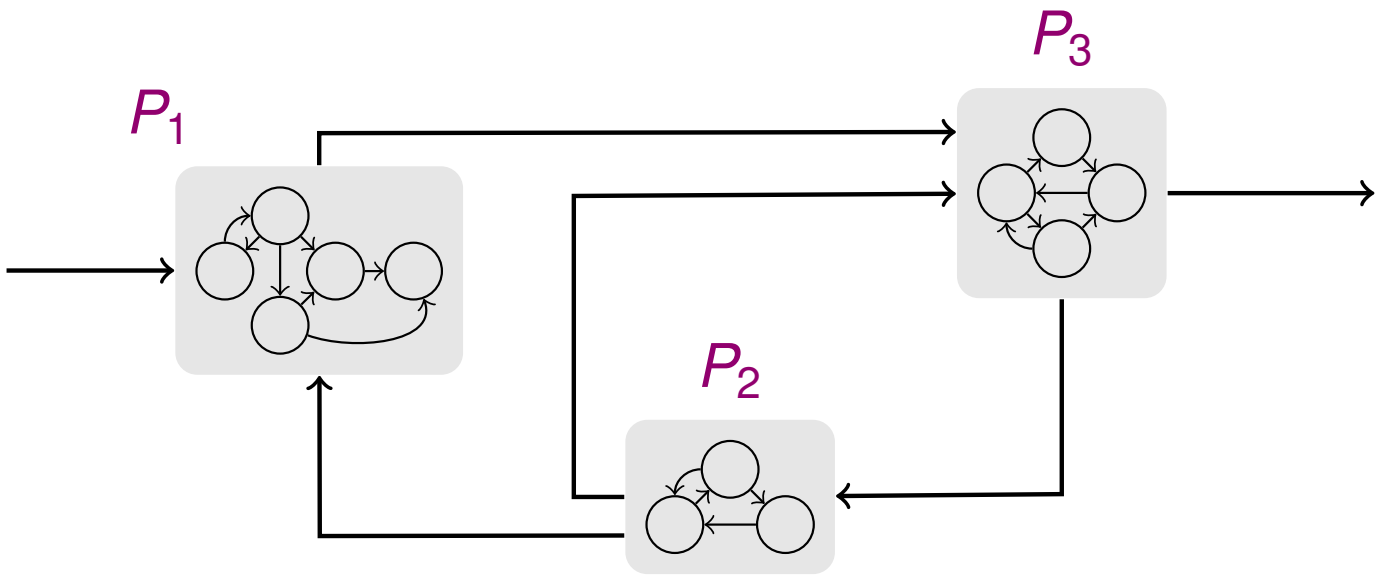
Programs interact with other programs and the physical world

Problem Overview



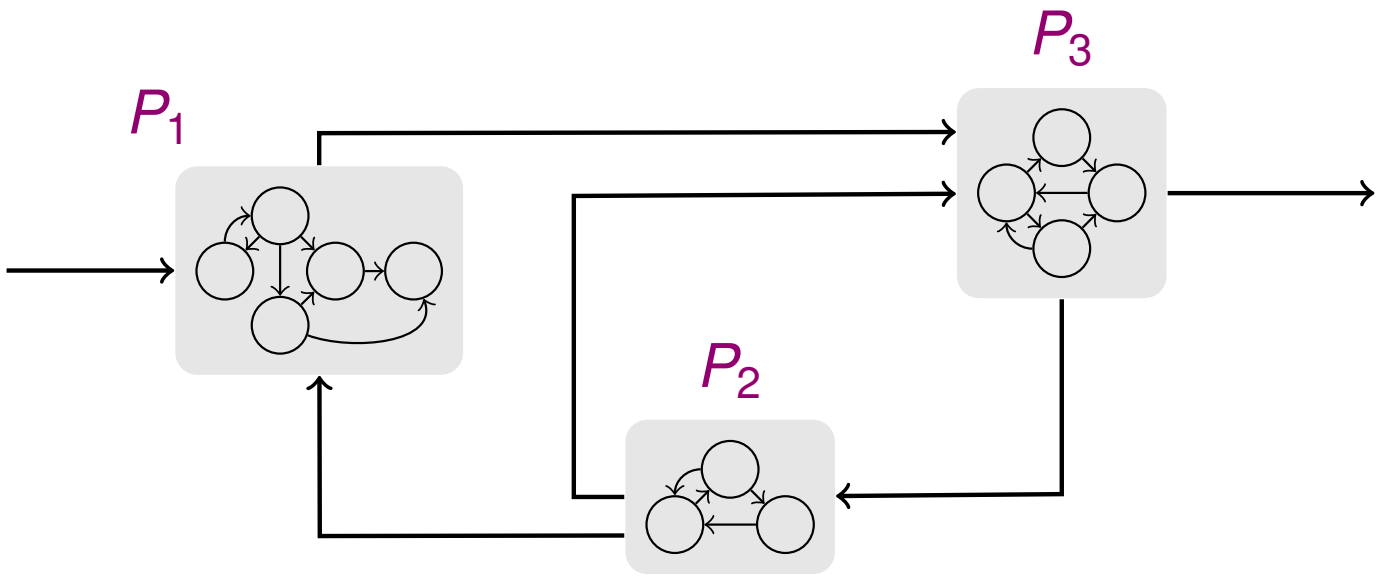
Predictability in the presence of uncertainty

Problem Overview



Small perturbation in input $\rightarrow$ Small perturbation in output

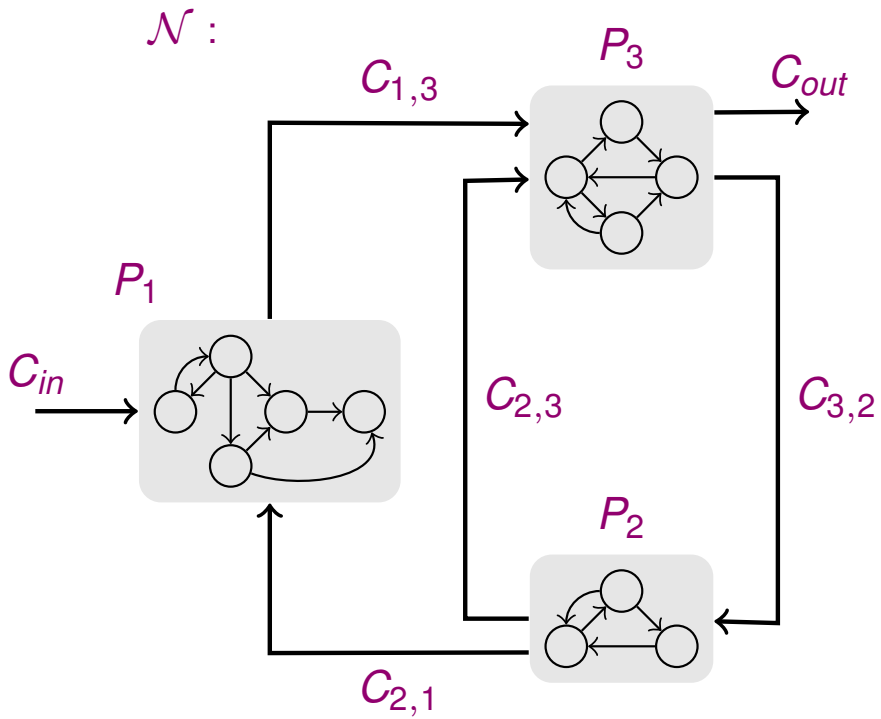
Problem Overview



Robustness!

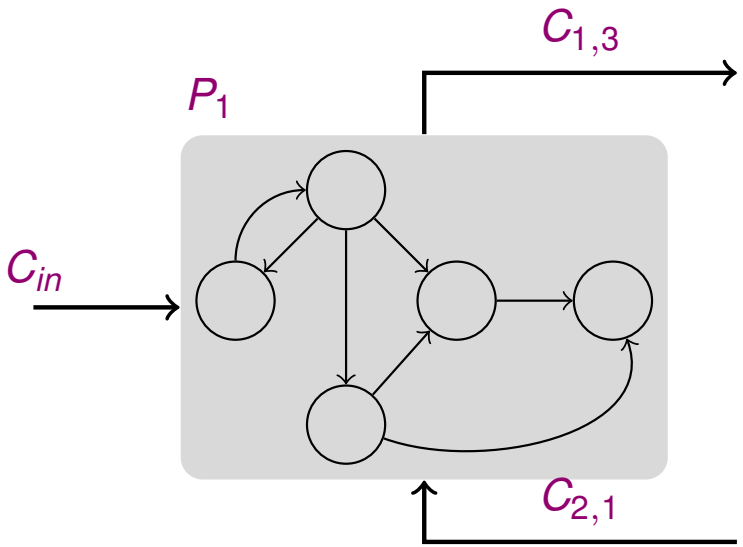
Synchronous Networked System

- Directed graph $(\mathcal{P}, \mathcal{C})$
  - $\mathcal{P} = \{P_1, \ldots, P_n\}$
  - $\mathcal{C} = I \cup N \cup O$
- Synchronous
- Computation alphabet: Σ



Process

- Process P_i: (In_i, Out_i, M_i)
- Mealy Machine M_i: $(\Sigma^{|In_i|}, \Sigma^{|Out_i|}, Q_i, q_{0_i}, R_i)$



Network Semantics

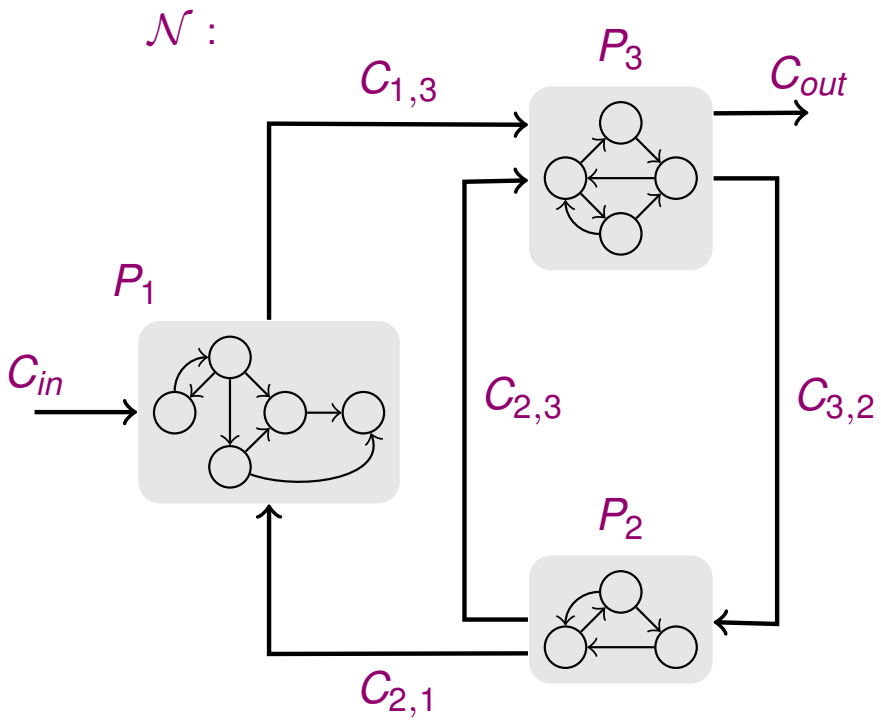
- $\mathcal{N}$: $(\Sigma^{|I|}, \Sigma^{|O|}, Q, \mathbf{q}_0, R)$
- Network state: $(q_1, \ldots, q_n, c_1, \ldots, c_{|N|})$
- Network transition:

$$(q_1, \ldots, q_n, c_1, \ldots, c_{|I|})$$

$$(a_1, \ldots, a_{|I|}), \quad (a'_1, \ldots, a'_{|w'|})$$

$$(q'_1, \ldots, q'_n, c'_1, \ldots, c'_{|O|})$$

- Network execution: $\rho(\mathbf{s})$
- Network output: $[\![\mathcal{N}]\!](\mathbf{s})$



Internal Channel Perturbations

- Instantaneous deletion or substitution of current symbol
- τ-transition:

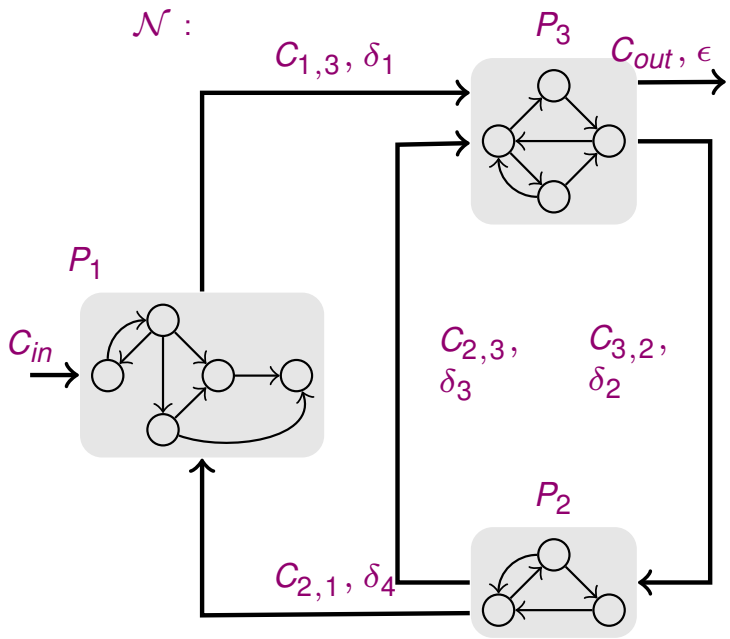
$$(q_1, \ldots, q_n, c_1, \ldots, c_{|I|})$$
$$\downarrow \varepsilon, \varepsilon$$
$$(q_1, \ldots, q_n, c'_1, \ldots, c'_{|I|}),$$

- Perturbed network execution: $\rho_\tau(\mathbf{s})$
- Perturbed network output: $[\![\rho_\tau]\!](\mathbf{s})$
- Channel-wise perturbation count in $\rho_\tau(\mathbf{s})$: $\|\rho_\tau(\mathbf{s})\|$

Robust Networked System

Given:

- a networked system $\mathcal{N}$,
- max. pertb. count of internal channels, $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_{|N|})$,
- max. permissible error in output channel, ϵ,
- distance metric d over $\Sigma^\star$



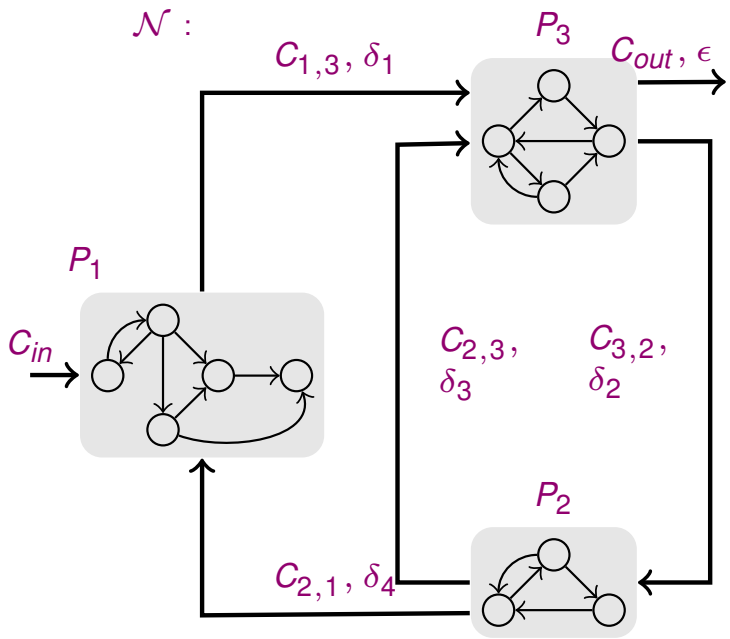
$\mathcal{N}$ is defined to be (δ, ϵ)-robust if:

$$\forall \mathbf{s} \in (\Sigma^{|I|})^\star, \forall \rho_\tau(\mathbf{s}) : \|\rho_\tau(\mathbf{s})\| \leq \delta \implies d(\llbracket \mathcal{N} \rrbracket(\mathbf{s}), \llbracket \rho_\tau \rrbracket(\mathbf{s})) \leq \epsilon$$

Robust Networked System

Given:

- a networked system $\mathcal{N}$,
- max. pertb. count of internal channels, $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_{|N|})$,
- max. permissible error in output channel, ϵ,
- distance metric d over $\Sigma^\star$



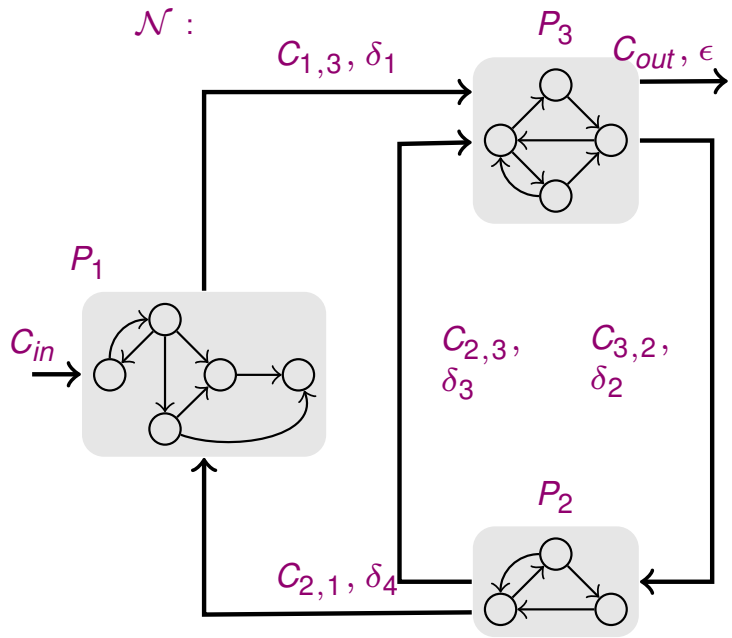
$\mathcal{N}$ is defined to be $(\boldsymbol{\delta}, \epsilon)$-robust if:

$$\forall \mathbf{s} \in (\Sigma^{|I|})^\star, \forall \rho_\tau(\mathbf{s}) : \ \|\rho_\tau(\mathbf{s})\| \leq \boldsymbol{\delta} \implies d(\ [\![\mathcal{N}]\!](\mathbf{s}), [\![\rho_\tau]\!](\mathbf{s})\ ) \leq \epsilon$$

Problem Definition

Given:

- a networked system $\mathcal{N}$,
- max. pertb. count of internal channels, $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_{|N|})$,
- max. permissible error in output channel, ϵ,
- distance metric d over $\Sigma^\star$

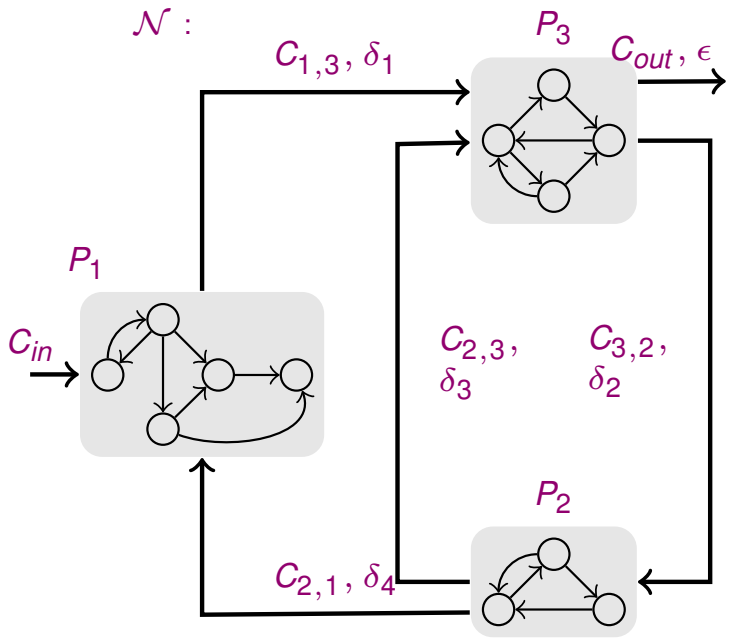


Check if $\mathcal{N}$ is $(\boldsymbol{\delta}, \epsilon)$-robust.

Solution Strategy

Given:

- a networked system $\mathcal{N}$,
- max. pertb. count of internal channels, $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_{|N|})$,
- max. permissible error in output channel, ϵ,
- distance metric d over $\Sigma^\star$



Construct machine $\mathcal{A}$: $\mathcal{N}$ is $(\boldsymbol{\delta}, \epsilon)$-robust iff $\mathcal{L}(\mathcal{A})$ is empty.

Solution Strategy

Construct machine $\mathcal{A}$: $\mathcal{N}$ is (δ, ϵ)-robust iff $\mathcal{L}(\mathcal{A})$ is empty.

- In a run of $\mathcal{A}$ on $\mathbf{s}$, $\mathcal{A}$ *simultaneously*:
  - simulates unperturbed, perturbed executions: $\rho(\mathbf{s})$, $\rho_\tau(\mathbf{s})$
  - tracks channel perturbations along $\rho_\tau(\mathbf{s})$,
  - tracks distance between outputs of $\mathcal{N}$ along $\rho(\mathbf{s})$, $\rho_\tau(\mathbf{s})$.
- $\mathcal{A}$ accepts $\mathbf{s}$ iff $\exists \rho_\tau(\mathbf{s})$: $\|\rho_\tau(\mathbf{s})\| \leq \delta$ *and* $d([\![\mathcal{N}]\!](\mathbf{s}), [\![\rho_\tau]\!](\mathbf{s})) > \epsilon$.

Review: Levenshtein Distance, $d_{Lev}(s, t)$

Minimum number of symbol *insertions*, *deletions* and *substitutions* required to transform s into t.

$d_{Lev}(abc, bcd) = 2$

$$\begin{array}{cccc} \sqcup & b & c & d \\ a & b & c & \sqcup \end{array}$$

$d_{Lev}(acbcd, ccff) = 4$

$$\begin{array}{cccccc} \sqcup & c & \sqcup & c & f & f \\ a & c & b & c & d & \sqcup \end{array} \text{ or,}$$

$$\begin{array}{ccccc} \sqcup & c & c & f & f \\ a & c & b & c & d \end{array}$$

Review: Levenshtein Distance, $d_{Lev}(s, t)$

Minimum number of symbol *insertions*, *deletions* and *substitutions* required to transform s into t.

$d_{Lev}(abc, bcd) = 2$

$\sqcup$	b	c	d
a	b	c	$\sqcup$

$d_{Lev}(acbcd, ccff) = 4$

$\sqcup$	c	$\sqcup$	c	f	f	or,
a	c	b	c	d	$\sqcup$	

$\sqcup$	c	c	f	f
a	c	b	c	d

Review: Levenshtein Distance, $d_{Lev}(s, t)$

Minimum number of symbol *insertions*, *deletions* and *substitutions* required to transform s into t.

$d_{Lev}(abc, bcd) = 2$

$\sqcup$	b	c	d
a	b	c	$\sqcup$

$d_{Lev}(acbcd, ccff) = 4$

$\sqcup$	c	$\sqcup$	c	f	f	or,
a	c	b	c	d	$\sqcup$	

$\sqcup$	c	c	f	f
a	c	b	c	d

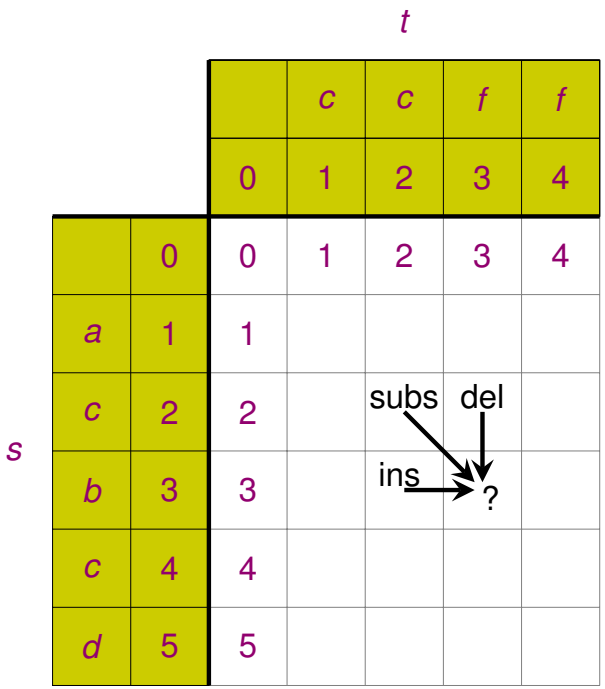
Review: Levenshtein Distance Computation

			t			
			c	c	f	f
		0	1	2	3	4
	0	0	1	2	3	4
a	1	1				
c	2	2				
b	3	3				
c	4	4				
d	5	5				

s

$$d_{Lev}(s[0], t[0]) = 0$$
$$d_{Lev}(s[0, i], t[0]) = i$$
$$d_{Lev}(s[0], t[0, j]) = j$$

Review: Levenshtein Distance Computation



$$d_{Lev}(s[0,i],\ t[0,j]) =$$
$$\min(d_{Lev}(s[0,i-1], t[0,j-1]) + \Delta(s[i], t[j]),$$
$$d_{Lev}(s[0,i-1],\ t[0,j]) + 1,$$
$$d_{Lev}(s[0,i],\ t[0,j-1]) + 1$$
$$)$$

$\Delta(a,b)$ equals 0 if $a = b$, else 1.

Review: Levenshtein Distance Computation

			t			
			c	c	f	f
		0	1	2	3	4
	0	0	1	2	3	4
a	1	1	1	2	3	4
c	2	2	1			
b	3	3	2			
c	4	4	3			
d	5	5	4			

s

$$
\begin{aligned}
&d_{Lev}(s[0,i],\ t[0,j]) = \\
&\min(d_{Lev}\,(s[0,i-1], t[0,j-1]) + \Delta(s[i], t[j]), \\
&\quad d_{Lev}\,(s[0,i-1],\ t[0,j]) + 1, \\
&\quad d_{Lev}\,(s[0,i],\ t[0,j-1]) + 1 \\
&\quad)
\end{aligned}
$$

$\Delta(a,b)$ equals 0 if $a = b$, else 1.

Review: Levenshtein Distance Computation

			t			
			c	c	f	f
		0	1	2	3	4
	0	0	1	2	3	4
a	1	1	1	2	3	4
c	2	2	1	1	2	3
b	3	3	2	2		
c	4	4	3	2		
d	5	5	4	3		

s

$$
\begin{aligned}
&d_{Lev}(s[0,i],\ t[0,j]) = \\
&\min(d_{Lev}\,(s[0,i-1], t[0,j-1]) + \Delta(s[i], t[j]), \\
&\qquad d_{Lev}\,(s[0,i-1],\ t[0,j]) + 1, \\
&\qquad d_{Lev}\,(s[0,i],\ t[0,j-1]) + 1 \\
&\qquad)
\end{aligned}
$$

$\Delta(a,b)$ equals 0 if $a = b$, else 1.

Review: Levenshtein Distance Computation

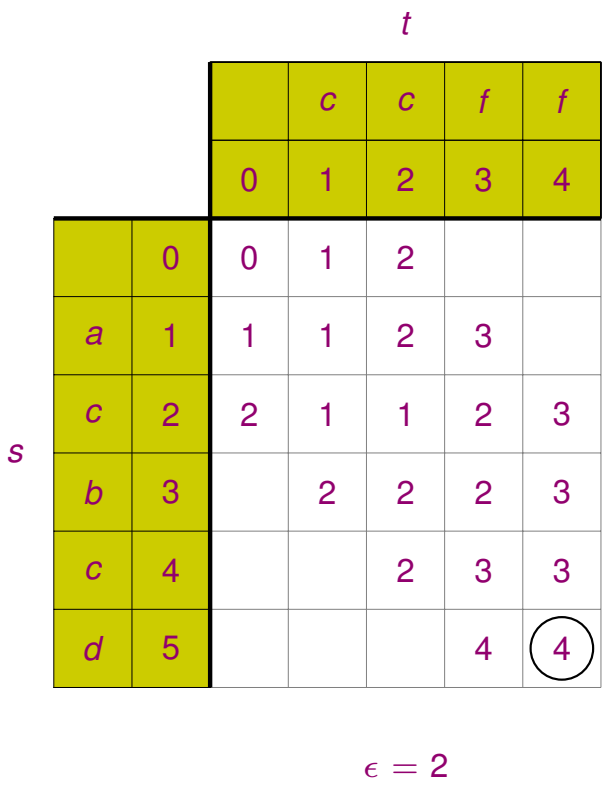
			t			
			c	c	f	f
		0	1	2	3	4
	0	0	1	2	3	4
a	1	1	1	2	3	4
c	2	2	1	1	2	3
b	3	3	2	2	2	3
c	4	4	3	2	3	3
d	5	5	4	3	4	4

s

$$
\begin{aligned}
&d_{Lev}(s[0,i],\ t[0,j]) = \\
&\min(d_{Lev}\,(s[0,i-1], t[0,j-1]) + \Delta(s[i], t[j]), \\
&\qquad d_{Lev}\,(s[0,i-1],\ t[0,j]) + 1, \\
&\qquad d_{Lev}\,(s[0,i],\ t[0,j-1]) + 1 \\
&\qquad)
\end{aligned}
$$

$\Delta(a,b)$ equals 0 if $a = b$, else 1.

Review: Levenshtein Distance Computation

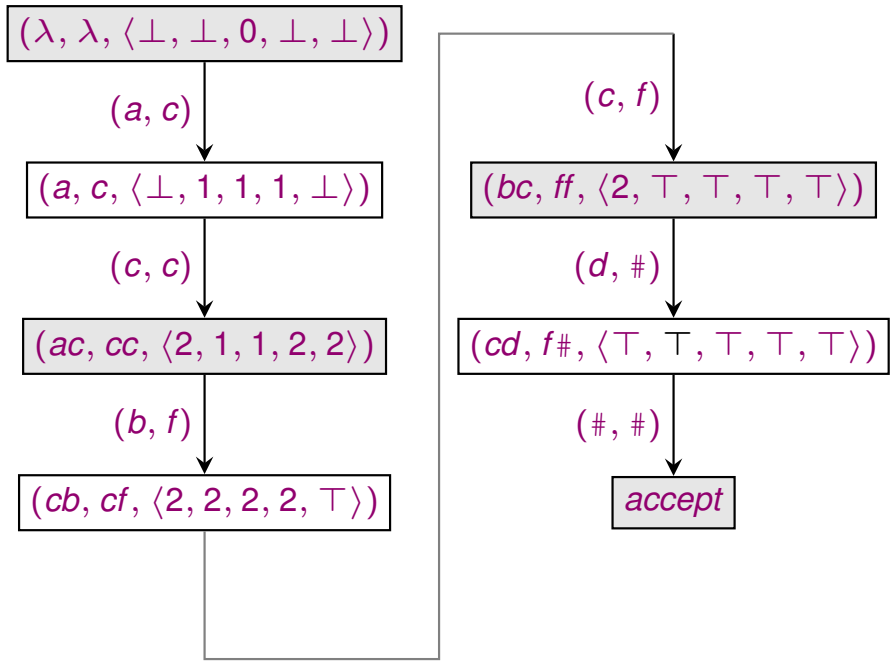


To check if $d_{Lev}(s,t) > \epsilon$, focus on ϵ-diagonal.

Construct DFA $\mathcal{D}^{\epsilon}_{Lev}$: runs on a string pair (s,t), and accepts iff $d_{Lev}(s,t) > \epsilon$.

Distance-Tracking Automaton, $\mathcal{D}^{\epsilon}_{Lev}$

			c	c	f	f	#	#
		0	1	2	3	4	5	6
	0	0	1	2				
a	1	1	1	2	$\top$			
c	2	2	1	1	2	$\top$		
b	3		2	2	2	$\top$	$\top$	
c	4			2	$\top$	$\top$	$\top$	$\top$
d	5				$\top$	$\top$	$\top$	$\top$
#	6					$\top$	$\top$	$\top$



$$\epsilon = 2$$

Distance-Tracking Automaton, $\mathcal{D}^{\epsilon}_{Lev}$

$\mathcal{D}^{\epsilon}_{Lev}$ accepts a pair of strings (s, t) iff $d_{Lev}(s, t) > \epsilon$.

Review: Counter Machines
(One-way, Nondeterministic)

h-counter machine $\mathcal{A}$: finite automaton with h integer counters

- In each step, $\mathcal{A}$ may
  - read an input symbol,
  - test counter values
  - change state
  - update each counter by some constant
- Transition: $(x, \sigma, test, x', c_1, \ldots, c_h)$
- s is accepted by $\mathcal{A}$ iff $(x_0, s_0, 0, \ldots 0) \rightarrow^\star_{\mathcal{A}} (acc, s_j, z_1, \ldots, z_h)$

Most interesting questions are undecidable.

Review: Counter Machines
(One-way, Nondeterministic)

h-counter machine $\mathcal{A}$: finite automaton with h integer counters

- In each step, $\mathcal{A}$ may
  - read an input symbol,
  - test counter values
  - change state
  - update each counter by some constant
- Transition: $(x, \sigma, test, x', c_1, \ldots, c_h)$
- s is accepted by $\mathcal{A}$ iff $(x_0, s_0, 0, \ldots 0) \rightarrow_{\mathcal{A}}^{\star} (acc, s_j, z_1, \ldots, z_h)$

Most interesting questions are undecidable.

Review: Counter Machines
(One-way, Nondeterministic)

h-counter machine $\mathcal{A}$: finite automaton with h integer counters

- In each step, $\mathcal{A}$ may
  - read an input symbol,
  - test counter values
  - change state
  - update each counter by some constant
- Transition: $(x, \sigma, test, x', c_1, \ldots, c_h)$
- s is accepted by $\mathcal{A}$ iff $(x_0, s_0, 0, \ldots 0) \rightarrow_{\mathcal{A}}^{\star} (acc, s_j, z_1, \ldots, z_h)$

Most interesting questions are undecidable.

Review: Counter Machines
(One-way, Nondeterministic)

h-counter machine $\mathcal{A}$: finite automaton with h integer counters

- In each step, $\mathcal{A}$ may
  - read an input symbol,
  - test counter values
  - change state
  - update each counter by some constant
- Transition: $(x, \sigma, test, x', c_1, \ldots, c_h)$
- s is accepted by $\mathcal{A}$ iff $(x_0, s_0, 0, \ldots 0) \rightarrow_{\mathcal{A}}^{\star} (acc, s_j, z_1, \ldots, z_h)$

Most interesting questions are undecidable.

Review: Reversal-bounded Counter Machines
(One-way, Nondeterministic)

r-reversal bounded, *h*-counter machine: each counter alternates between an increasing mode and a decreasing mode at most r times.

[GurariIbarra1981]
Nonemptiness for a *r*-reversal bounded, *h*-counter machine $\mathcal{A}$ is solvable in time polynomial in the size of $\mathcal{A}$.

Review: Reversal-bounded Counter Machines
(One-way, Nondeterministic)

r-reversal bounded, h-counter machine: each counter alternates between an increasing mode and a decreasing mode at most r times.

[GurariIbarra1981]
Nonemptiness for a r-reversal bounded, h-counter machine $\mathcal{A}$ is solvable in time polynomial in the size of $\mathcal{A}$.

Robustness Analysis for Levenshtein distance

Construct machine $\mathcal{A}_{Lev}^{\delta,\epsilon} \mathcal{A}$: $\mathcal{N}$ is (δ, ϵ)-robust iff $\mathcal{L}(\mathcal{A})$ is empty.

- In a run of $\mathcal{A}$ on $\mathbf{s}$, $\mathcal{A}$ *simultaneously*:
  - simulates unperturbed, perturbed executions: $\rho(\mathbf{s})$, $\rho_\tau(\mathbf{s})$
  - tracks channel perturbations along $\rho_\tau(\mathbf{s})$,
  - tracks distance between outputs of $\mathcal{N}$ along $\rho(\mathbf{s})$, $\rho_\tau(\mathbf{s})$.
- $\mathcal{A}$ accepts $\mathbf{s}$ iff $\exists \rho_\tau(\mathbf{s})$: $\|\rho_\tau(\mathbf{s})\| \leq \delta$ *and* $d([\![\mathcal{N}]\!](\mathbf{s}), [\![\rho_\tau]\!](\mathbf{s})) > \epsilon$.

$\mathcal{A}$: 1-reversal-bounded $|N|$-counter machine

- A state $\mathbf{x}$: $(\mathbf{q}, \mathbf{r}, \mathbf{q}_{Lev})$, $\mathbf{q}, \mathbf{r} \in Q$, $\mathbf{q}_{Lev} \in Q_{Lev}$
- Initial state: $(\mathbf{q}_0, \mathbf{q}_0, \mathbf{q}_{0\,Lev})$
- Set of counters $Z = \{z_1, \ldots, z_{|N|}\}$, initially 0
- Final state: $\{\mathbf{acc}\}$

$\mathcal{A}$: Initialization transition

Initialize counters with perturbation bounds:

$$\left((\mathbf{q}_0, \mathbf{q}_0, \mathbf{q}_{0\,Lev}), \quad \varepsilon, \quad \bigwedge_k z_k = 0, \quad (\mathbf{q}_0, \mathbf{q}_0, \mathbf{q}_{0\,Lev}), \quad (+\delta_1, \ldots, +\delta_{|N|}) \right)$$

$\mathcal{A}$: Unperturbed network transitions

Simulate pair of unperturbed $\mathcal{N}$-transitions, track output distance:

$$\left((\mathbf{q}, \mathbf{r}, \mathbf{q}_{Lev}), \quad \mathbf{a}, \quad \bigwedge_k z_k \geq 0, \quad (\mathbf{q}', \mathbf{r}', \mathbf{q}'_{Lev}), \quad \mathbf{0} \right)$$

$\mathcal{A}$: Perturbed network transitions

Simulate τ-transition, decrement counters of perturbed channels by 1:

$$\left((\mathbf{q}, \mathbf{r}, \mathbf{q}_{Lev}), \quad \varepsilon, \quad \bigwedge_k z_k \geq 0, \quad (\mathbf{q}, \mathbf{r}_\tau, \mathbf{q}_{Lev}), \quad \mathbf{c} \right)$$

$\mathcal{A}$: Rejecting transitions

Reject if any internal channel perturbation exceeds bound:

$$\left((\mathbf{q}, \mathbf{r}, \mathbf{q}_{Lev}), \quad \varepsilon, \quad \bigvee_k z_k < 0, \quad \mathbf{rej}, \quad \mathbf{0} \right)$$

$$(\mathbf{rej}, \mathbf{a}, \mathit{true}, \mathbf{rej}, \mathbf{0})$$

$\mathcal{A}$: Accepting transitions

Accept witness to non-robustness:

$$\left((\mathbf{q}, \mathbf{r}, acc_{Lev}), \quad \varepsilon, \quad \bigwedge_k z_k \geq 0, \quad \mathbf{acc}, \quad \mathbf{0} \right)$$

Main Result(s)

Checking if $\mathcal{N}$ is (δ, ϵ)-robust w.r.t the Levenshtein distance is polynomial in the size of $\mathcal{N}$ and exponential in ϵ.

Checking if $\mathcal{N}$ is (δ, ϵ)-robust w.r.t the L_1-norm is polynomial in the size of $\mathcal{N}$.

Main Result(s)

Checking if $\mathcal{N}$ is (δ, ϵ)-robust w.r.t the Levenshtein distance is polynomial in the size of $\mathcal{N}$ and exponential in ϵ.

Checking if $\mathcal{N}$ is (δ, ϵ)-robust w.r.t the L_1-norm is polynomial in the size of $\mathcal{N}$.

Related Work

- Sequential programs with perturbed inputs [MS09, CGLN11]
- Input-output stability of finite-state transducers [TBCSM12]
- Sequential circuits, *common suffix distance metric* [DHLN10]
- Robust control, wireless control networks [ADJPW11, Pappas11]
- Reactive systems with ω-regular spec. in uncertain environment [MRT11, CHR10, BGHJ09]

Future Work

- Consider more distance-metrics
- Generalize error model - input noise, process failures etc.
- Synthesis of robust networked systems

Thank you.