Youssef Hanna
Iowa State U.

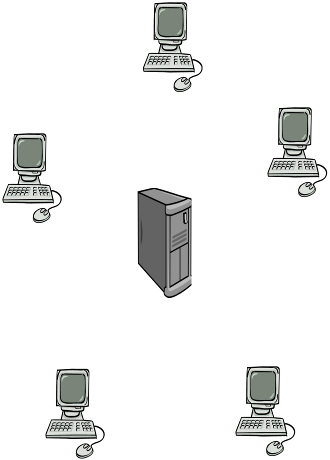
Samik Basu
Iowa State U.

Hridesh Rajan
Iowa State U.

Behavioral Automata Composition for Automatic Topology Independent Verification of Parameterized Systems

Midwest Verification Day 2009

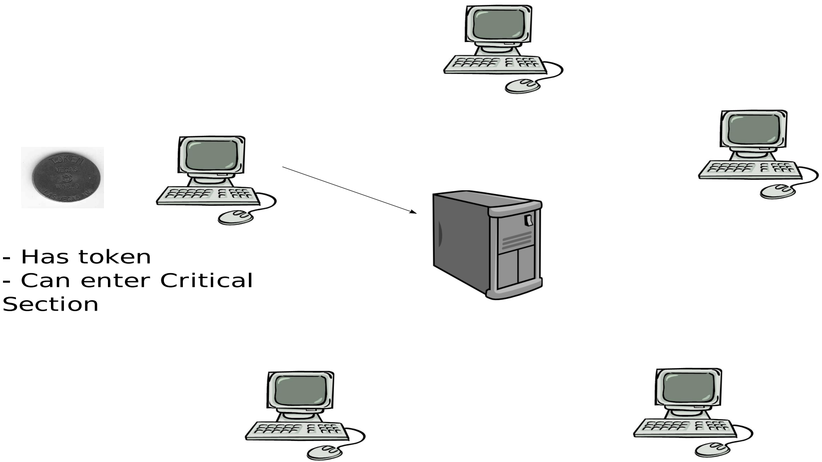
What is a Parameterized System?



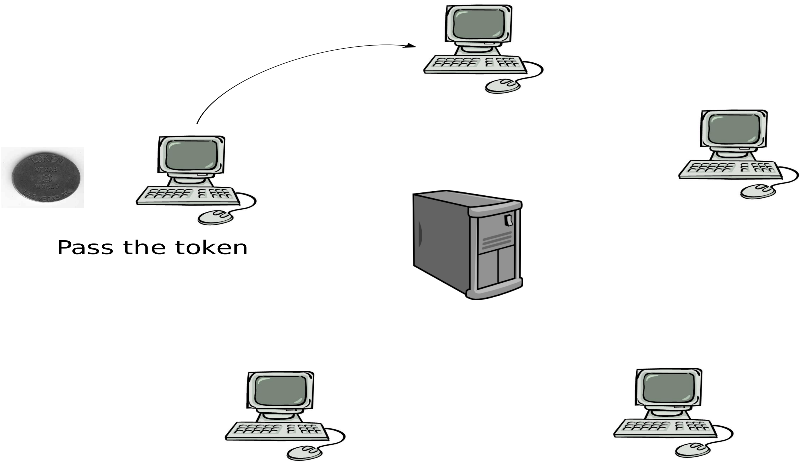
- A system with *n* homogeneous processes.
- Example:
  - Distributed Mutual Exclusion [a].
  - n = 5.
  - **Goal**: Each process to access the shared resource exclusively.

[a] Wolper, P. and Lovinfosse, V. 1990. Verifying properties of large sets of processes with network invariants. In J. Sifakis (ed), *Automatic Verification Methods For Finite State Systems*. Springer-Verlag, LNCS 407.

Distributed Mutual Exclusion

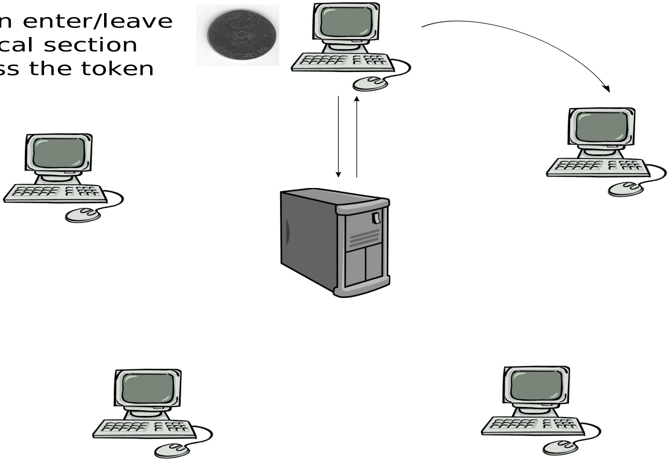


Distributed Mutual Exclusion

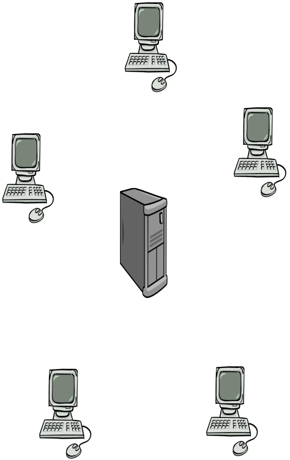


Distributed Mutual Exclusion

- Can enter/leave critical section
- Pass the token



Verifying Parameterized Systems



- Example property to verify $\varphi(\boldsymbol{i}, \boldsymbol{j})$:
  - no 2 processes ***i*** and ***j*** in Critical Section concurrently.
- Property satisfied for *this* system where $n = 5$
- Is $\varphi(\boldsymbol{i}, \boldsymbol{j})$ satisfied for $n = 6, 7, \ldots$?

Verifying Parameterized Systems

- **Problem of Verifying Parameterized Systems:**
  Given a parameterized system $\boldsymbol{sys(n)}$ and a property φ, is the property satisfied for every instance of the system $(\forall \boldsymbol{n} : \boldsymbol{sys(n)} \models \varphi)$?
- This is an undecidable problem [1].

[1] K. R. Apt and D. C. Kozen. Limits for automatic verification of finite-state con- current systems. Inf. Process. Lett., 22(6):307-309, 1986.

Existing Work

- Large amount of existing work[2] [3].
- Key idea based on the notion of *cut-off*.
- Identify k s.t.

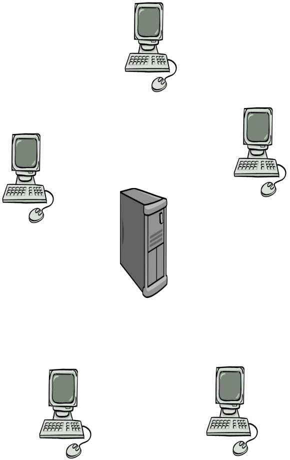
$$sys(k) \models \varphi \Leftrightarrow \forall n > k : sys(n) \models \varphi$$

- No need to verify properties for $n > k$
- k is called the *cut-off*

[2] E. A. Emerson, R. J. Trefler, and T. Wahl. Reducing model checking of the few to the one. In ICFEM, pp. 94-113, 2006.

[3] C. N. Ip and D. L. Dill. Verifying systems with replicated components in murphi. In CAV, pages 147-158, 1996.

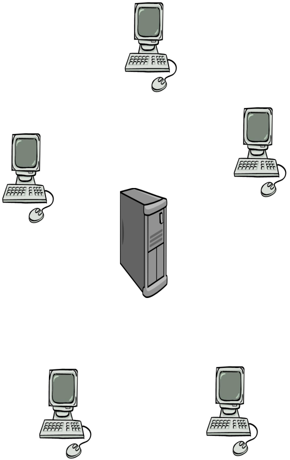
Existing Work



- Emerson and Namjoshi found $\boldsymbol{k} = 4$ for networks with ring topology for properties $\varphi(\boldsymbol{i}, \boldsymbol{j})_{\boldsymbol{i} \neq \boldsymbol{j}}$ [a].

[a] E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In POPL, pages 85-94, 1995

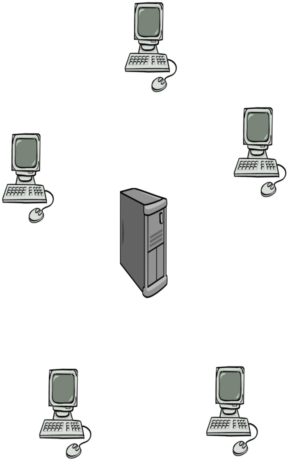
Existing Work



- Emerson and Namjoshi found $\boldsymbol{k} = 4$ for networks with ring topology for properties $\varphi(\boldsymbol{i},\boldsymbol{j})_{\boldsymbol{i}\neq\boldsymbol{j}}$ [a].
- For $\varphi(\boldsymbol{i},\boldsymbol{j})$: no 2 processes $\boldsymbol{i}$ and $\boldsymbol{j}$ in Critical Section concurrenty:
  - $\boldsymbol{sys}(4) \models \varphi(\boldsymbol{i},\boldsymbol{j}) \Leftrightarrow \forall \boldsymbol{n} > \boldsymbol{k} : \boldsymbol{sys}(\boldsymbol{n}) \models \varphi(\boldsymbol{i},\boldsymbol{j})$

[a] E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In POPL, pages 85-94, 1995

Existing Work



- Emerson and Namjoshi found $\boldsymbol{k} = 4$ for networks with ring topology for properties $\varphi(\boldsymbol{i}, \boldsymbol{j})_{\boldsymbol{i} \neq \boldsymbol{j}}$ [a].
- For $\varphi(\boldsymbol{i}, \boldsymbol{j})$: no 2 processes $\boldsymbol{i}$ and $\boldsymbol{j}$ in Critical Section concurrenty:
  - $\boldsymbol{sys}(4) \models \varphi(\boldsymbol{i}, \boldsymbol{j}) \Leftrightarrow \forall \boldsymbol{n} > \boldsymbol{k}$: $\boldsymbol{sys}(\boldsymbol{n}) \models \varphi(\boldsymbol{i}, \boldsymbol{j})$
- No need to verify properties of the form $\varphi(\boldsymbol{i}, \boldsymbol{j})$ for $\boldsymbol{n} > 4$.

[a] E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In POPL, pages 85-94, 1995

Problem with Existing Work

- Most ideas focus on a specific system
  - e.g. for ring systems and property φ, cut-off = $\boldsymbol{k}$.

Problem with Existing Work

- Most ideas focus on a specific system
  - e.g. for ring systems and property φ, cut-off = $\boldsymbol{k}$.
  - Not immediately applicable to new systems
  - ...without developing new theories from first principles.

Problem with Existing Work

- Most ideas focus on a specific system
  - e.g. for ring systems and property φ, cut-off = $\boldsymbol{k}$.
  - Not immediately applicable to new systems
  - ...without developing new theories from first principles.
- System behavior not considered in cut-off computation
  - e.g. systems organized in a ring topology.

Problem with Existing Work

- Most ideas focus on a specific system
  - e.g. for ring systems and property φ, cut-off = $\boldsymbol{k}$.
  - Not immediately applicable to new systems
  - . . . without developing new theories from first principles.
- System behavior not considered in cut-off computation
  - e.g. systems organized in a ring topology.
  - \+ Generic cut-off, applies to other systems organized as ring
  - Computed cut-off value is often larger (i.e. more nodes).
  - More nodes $\Rightarrow$ Larger verification models.
  - Larger models $\Rightarrow$ Increased verification cost

A Programming Language Design Perspective

Can we represent parameterized systems in a manner, which enables automatic computation of the cut-off value?

Yes, We Can!

Language-based Technique for Cut-off Computation

Technical Contributions:

A representation strategy to specify system as input,
which enables
a novel cut-off generation algorithm.

Key Benefits:

- Fully automated – Ease of verification, *no PhD required*.
- Topology independent – User can specify the topology.
- Protocol-specific cut-off – often tighter i.e. reduced costs.

Key Idea

1. Given all possible actions of a process, system topology

Key Idea

1. Given all possible actions of a process, system topology
2. If we can generate the maximum behavior of that process in *any* environment, and

Key Idea

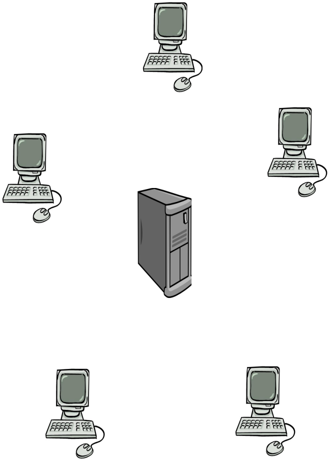
1. Given all possible actions of a process, system topology
2. If we can generate the maximum behavior of that process in *any* environment, and
3. Find smallest system that allows any one process to exhibit its maximum behavior, then **the size of this network is the cut-off.**

Key Idea

1. Given all possible actions of a process, system topology
2. If we can generate the maximum behavior of that process in *any* environment, and
3. Find smallest system that allows any one process to exhibit its maximum behavior, then **the size of this network is the cut-off.**

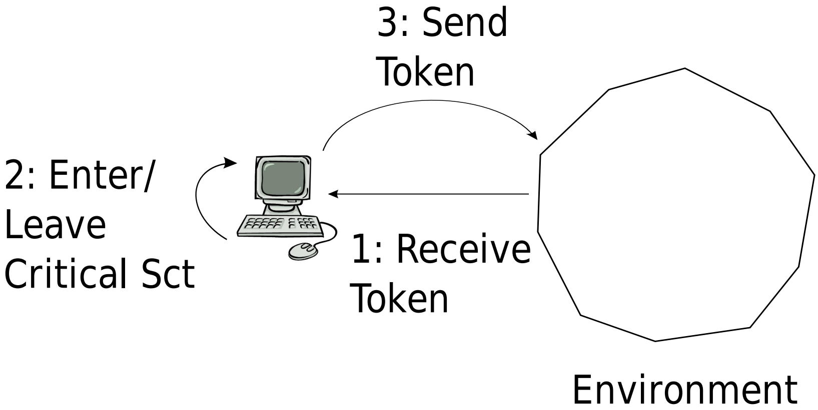
- *Any larger system cannot exhibit any more behavior.*

Define Process Actions

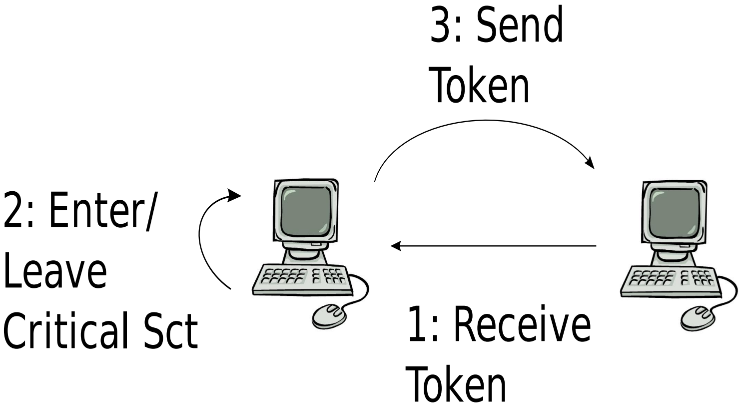


- **Process Actions:**
  - Receive token.
  - Enter/Leave critical section.
  - Send token.
- **Topology**
  - Ring.

Generate Behavior of one process in Any Environment



Find Smallest System Exhibiting Full Process Behavior



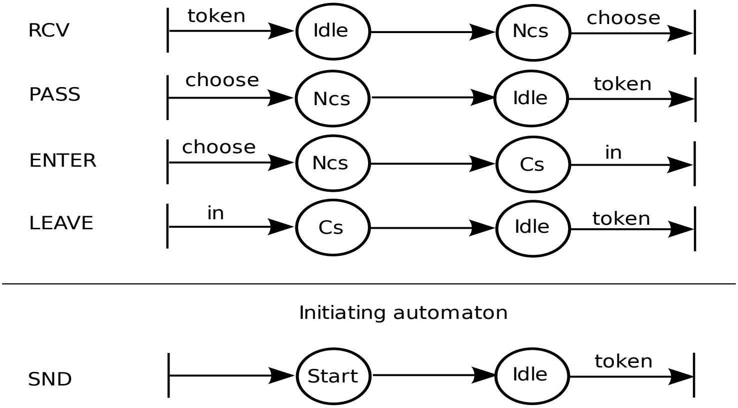
Outline

- Parameterized System Definition
- Problem Statement
- **Solution:**
  - Define all possible actions of a process.
  - Define system topology, system start configuration.
  - Generate maximum behavior of a process in *any* environment.
  - Find smallest system allowing such behavior to be exhibited.
- Case Studies

Steps for Our Technique

1. **Define actions of a process**
2. Define system topology and start configuration
3. Generate maximum behavior of one process in any environment
4. Find smallest system that allows any one process to perform its maximum behavior

Behavioral Automata: Atomic Actions

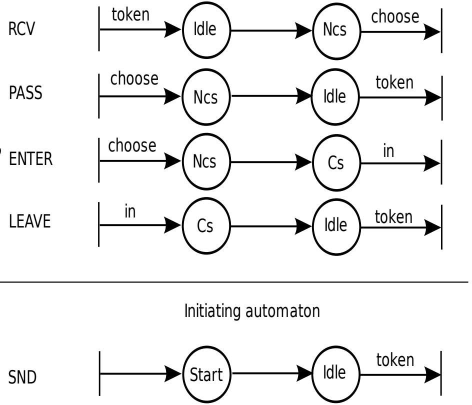


Steps for Our Technique

1. Define actions of a process
2. **Define system topology and start configuration**
3. Generate maximum behavior of one process in any environment
4. Find smallest system that allows any one process to perform its maximum behavior

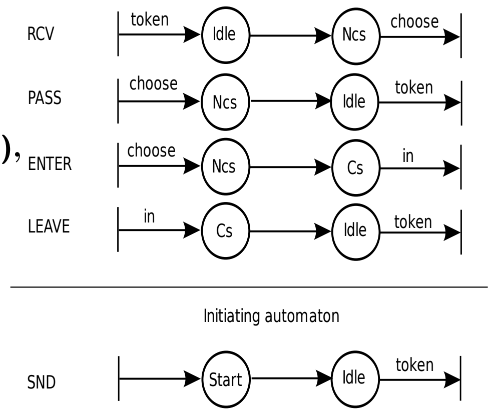
System Topology

- Topology = $\{(\mathtt{token}, i, (i+1) \bmod k), (\mathtt{in}, i, i), (\mathtt{choose}, i, i)\}$



System Topology

- Topology = $\{(\texttt{token}, i, (i+1) \bmod k), (\texttt{in}, i, i), (\texttt{choose}, i, i)\}$
- Start Configuration
  - 1 : SND
  - rest: RCV

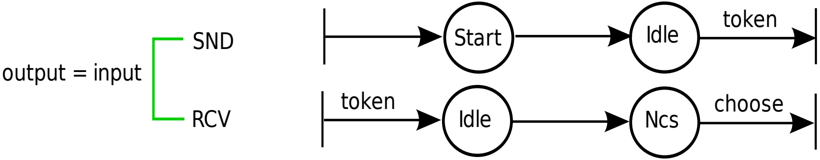


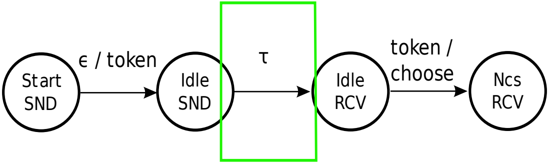
Steps for Our Technique

1. Define actions of a process
2. Define system topology and start configuration
3. **Generate maximum behavior of one process in any environment**
4. Find smallest system that allows any one process to perform its maximum behavior

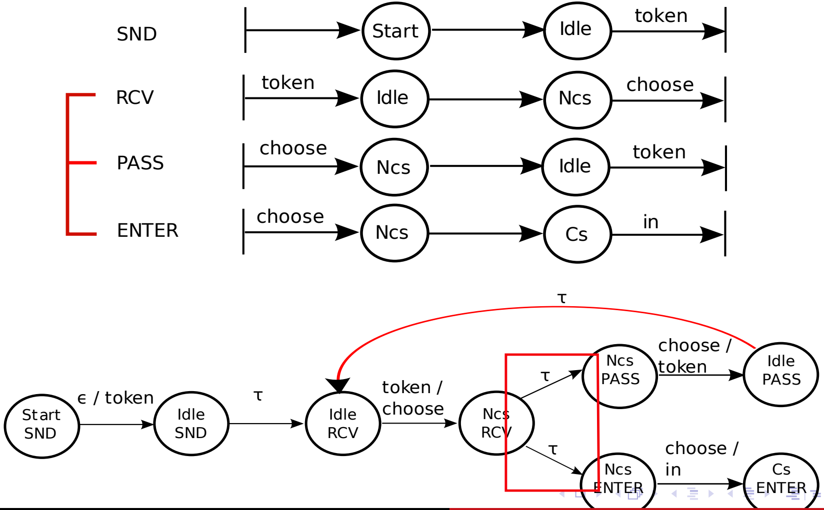
Behavior of One Process in Any Environment

Done by Composing behavioral automata

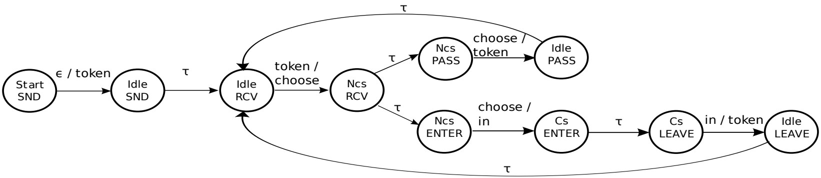




Behavior of One Process in Any Environment



Behavior of One Process in Any Environment

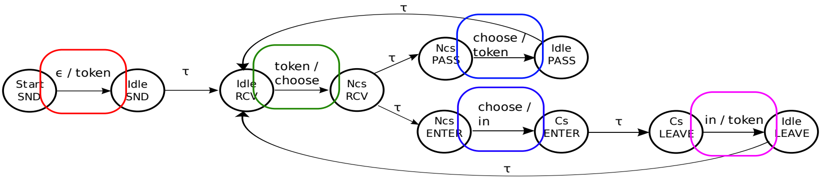


Steps for Our Technique

1. Define actions of a process
2. Define system topology and start configuration
3. Generate maximum behavior of one process in any environment
4. **Find smallest system that allows any one process to perform its maximum behavior**

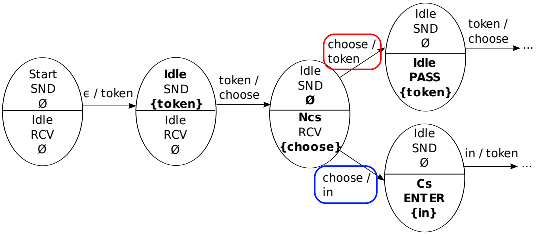
Goal of Finding Smallest Network

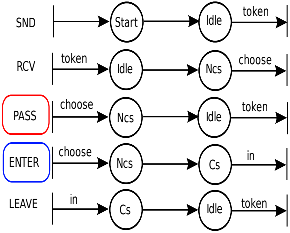
- Find smallest network that allows any process to exhibit its maximum behavior. The size of this network (***k***) is the cut-off.



$$\boldsymbol{sys(k)} \models \varphi \Leftrightarrow \forall \boldsymbol{n} > \boldsymbol{k} : \boldsymbol{sys(n)} \models \varphi$$

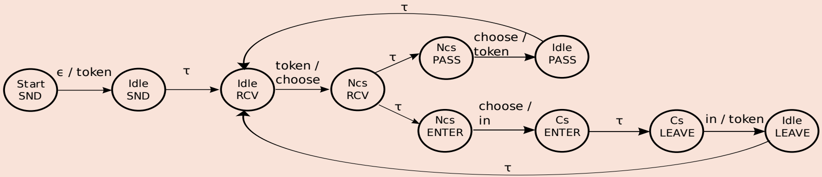
System with 2 processes sys(2)



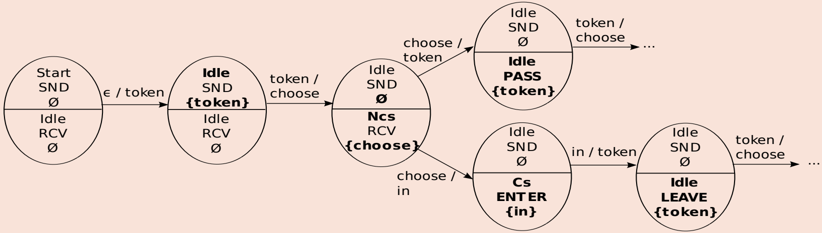


- Topology = $\{(\texttt{token}, \boldsymbol{i}, (\boldsymbol{i} + 1) \bmod \boldsymbol{k}), (\texttt{in}, \boldsymbol{i}, \boldsymbol{i}), (\texttt{choose}, \boldsymbol{i}, \boldsymbol{i})\}$

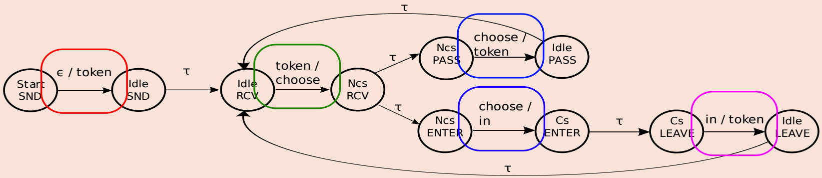
Behavior of one process in any environment



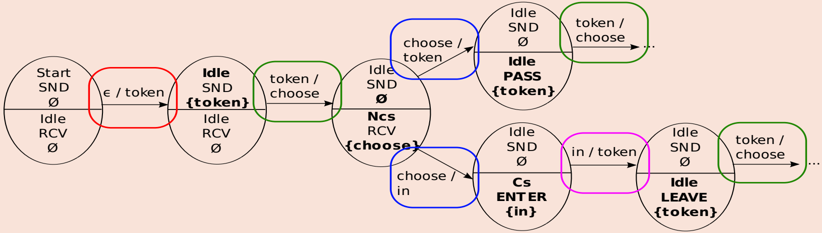
System with 2 processes



Behavior of one process in any environment



System with 2 processes



- A system with **2** processes allows one process to exhibit its maximum behavior
- **cut-off for DME is k = 2**
- $\boldsymbol{sys}(\mathbf{2}) \models \varphi \Leftrightarrow \forall \boldsymbol{n} > \mathbf{2} : \boldsymbol{sys}(\mathbf{2}) \models \varphi$

Kind of Properties φ

Properties involving one process i

$$\forall i, 1 \leq i \leq k : sys(k) \models \varphi(i) \Leftrightarrow \forall n \geq k, \forall i, 1 \leq i \leq n : sys(n) \models \varphi(i)$$

Properties involving 2 interdependent processes i, j

$$\forall i, j, 1 \leq i, j \leq k, i \neq j : sys(k) \models \varphi(i, j) \Leftrightarrow \forall n \geq k, \forall i, j, 1 \leq i, j \leq n, i \neq j : sys(n) \models \varphi(i, j)$$

Comparison with Other Techniques

	Topology	Existing Work		Our Technique	
		Cut-off	φ	Cut-off	φ
Dining Philosophers	Ring	5 [d]	i, j	3	i, j
Spin Lock	Star	3 [e]	i, j	2	i, j

[d] E. A. Emerson and V. Kahlon. Model checking large-scale and parameterized resource allocation systems. In TACAS, pages 251-265, 2002.

[e] S. Basu and C. R. Ramakrishnan. Compositional analysis for verification of parameterized systems. Theor. Comput. Sci., 354(2):211-229, 2006.

Conclusion

- Parameterized system verification is important.
- Reduce problem to verify small system with cut-off k.
- Current solutions are topology specific.
  - New theories required for every new system.
- **Our Solution:**
  - Automated technique for cut-off generation.
  - Topology Independent.
- **Future Work:** Apply technique to
  - Synchronous systems.
  - Infinite-data domain systems.

Questions?

`http://www.cs.iastate.edu/~slede/`

System with 2 processes sys(2)

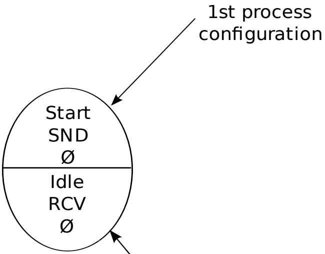
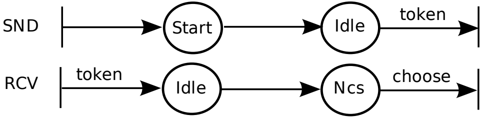


Figure: Start state in sys(2)



- A configuration consists of
  - The state of the process
  - The automaton of the process
  - The set of its output messages to be consumed

System with 2 processes sys(2)

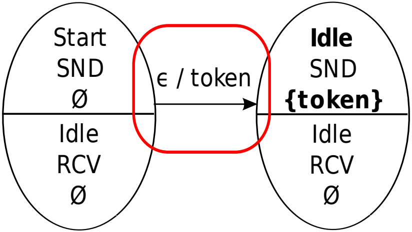
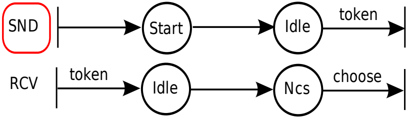


Figure: Autonomous action



System with 2 processes sys(2)

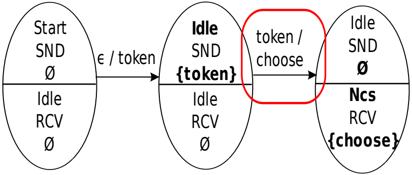
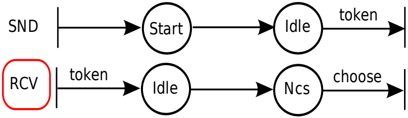
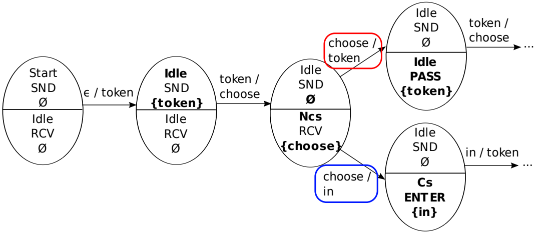


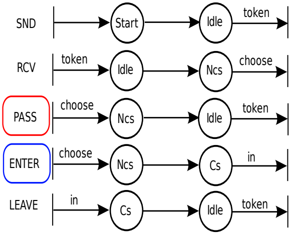
Figure: Non-autonomous Action



- Topology = $\{(\texttt{token}, \boldsymbol{i}, \boldsymbol{i} + \mathbf{1}),$ $(\texttt{in}, \boldsymbol{i}, \boldsymbol{i}),$ $(\texttt{choose}, \boldsymbol{i}, \boldsymbol{i})\}$

System with 2 processes sys(2)





- Topology = $\{(\texttt{token}, \boldsymbol{i}, \boldsymbol{i}+1), (\texttt{in}, \boldsymbol{i}, \boldsymbol{i}), (\texttt{choose}, \boldsymbol{i}, \boldsymbol{i})\}$

Proof of Soundness

- Assume smallest system $\boldsymbol{sys(k)}$ simulating full process behavior is not the cut-off
- $\boldsymbol{sys(n)} \models \varphi(\boldsymbol{i})$ while $\boldsymbol{sys(k)} \not\models \varphi(\boldsymbol{i})$ for $\boldsymbol{n} > \boldsymbol{k}$

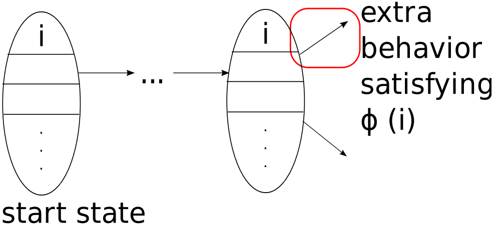


Figure: sys(n)

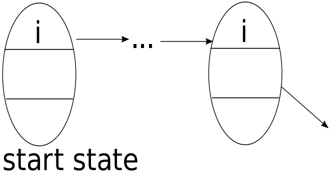


Figure: sys(k)

Proof of Soundness

- To verify property $\varphi(\boldsymbol{i})$ we **ONLY** care for process ***i***
- All processes are homogeneous
- $\boldsymbol{sys(k)}$ covers all possible behavior of any process (***i***) in any environment (***j***)

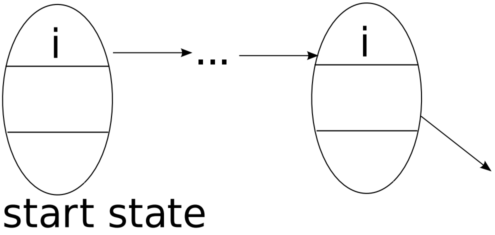


Figure: sys(k)

Proof of Soundness

- **For any $n > k$, no such transition is possible since all such transitions belong to behavior of i-th process which are covered by $sys(k)$**

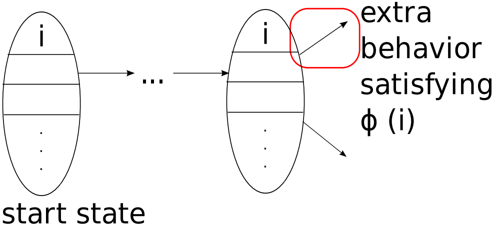


Figure: sys(n)