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.

Distributed Mutual Exclusion

- Has token
- Can enter Critical Section
Distributed Mutual Exclusion

Pass the token
Distributed Mutual Exclusion

- Can enter/leave critical section
- Pass the token
Verifying Parameterized Systems

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

Given a parameterized system \( \text{sys}(n) \) and a property \( \varphi \), is the property satisfied for every instance of the system

\[
\left( \forall n : \text{sys}(n) \models \varphi \right)
\]

This is an undecidable problem \(^1\).

---

Existing Work

- Large amount of existing work\(^2\) \(^3\).
- Key idea based on the notion of *cut-off*.
- Identify \(k\) s.t.
  \[
  \text{sys}(k) \models \varphi \iff \forall n > k : \text{sys}(n) \models \varphi
  \]
- No need to verify properties for \(n > k\)
- \(k\) is called the *cut-off*

---


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

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

For $\varphi(i, j)$: no 2 processes $i$ and $j$ in Critical Section concurrently:

$\text{sys}(4) \models \varphi(i, j) \iff \forall n > k : \text{sys}(n) \models \varphi(i, j)$

\textsuperscript{a}E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In POPL, pages 85-94, 1995
Emerson and Namjoshi found $k = 4$ for networks with ring topology for properties $\varphi(i, j)_{i \neq j}$.

For $\varphi(i, j)$: no 2 processes $i$ and $j$ in Critical Section concurrently:

- $\text{sys}(4) \models \varphi(i, j) \iff \forall n > k : \text{sys}(n) \models \varphi(i, j)$

No need to verify properties of the form $\varphi(i, j)$ for $n > 4$.

---

Most ideas focus on a specific system

- e.g. for ring systems and property $\varphi$, cut-off $= k$. 

- 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
Problem with Existing Work

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

1: Receive Token

2: Enter/Leave Critical Sect

3: Send Token

Environment
Find Smallest System Exhibiting Full Process Behavior

1: Receive Token
2: Enter/Leave Critical Sect
3: Send Token
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

- RCV: $\text{RCV} \xrightarrow{\text{token}} \text{Idle} \xrightarrow{\text{choose}} \text{Ncs}$
- PASS: $\text{PASS} \xrightarrow{\text{choose}} \text{Ncs} \xrightarrow{\text{token}} \text{Idle}$
- ENTER: $\text{ENTER} \xrightarrow{\text{choose}} \text{Ncs} \xrightarrow{\text{in}} \text{Cs} \xrightarrow{\text{in}} \text{Idle}$
- LEAVE: $\text{LEAVE} \xrightarrow{\text{in}} \text{Cs} \xrightarrow{\text{in}} \text{Idle}$

Initiating automaton:
- SND: $\text{SND} \xrightarrow{\text{token}} \text{Start} \xrightarrow{\text{token}} \text{Idle}$
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 = \{(\text{token}, i, (i + 1) \mod k), (	ext{in}, i, i), (\text{choose}, i, i)\}

- **RCV**: 
  - token → Idle → Ncs → choose

- **PASS**: 
  - choose → Ncs → Idle → token

- **ENTER**: 
  - choose → Ncs → Cs → in

- **LEAVE**: 
  - in → Cs → Idle → token

- **SND**: 
  - Start → Idle → token

Initiating automaton
System Topology

- Topology = \{(\text{token}, i, (i + 1) \mod k), (\text{in}, i, i), (\text{choose}, i, i)\}

- Start Configuration
  - 1 : SND
  - rest: RCV

Initiating automaton

- RCV
  - token \rightarrow \text{Idle} \rightarrow \text{Ncs} \rightarrow \text{choose}
- PASS
  - choose \rightarrow \text{Ncs} \rightarrow \text{Idle} \rightarrow \text{token}
- ENTER
  - choose \rightarrow \text{Ncs} \rightarrow \text{Cs} \rightarrow \text{in}
- LEAVE
  - in \rightarrow \text{Cs} \rightarrow \text{Idle} \rightarrow \text{token}

Laboratory for Software Design, Iowa State University

Automatic Cut-off Generation for Parameterized Systems
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

output = input

SND

RCV

Start → Idle

Idle → Ncs

token / choose

SND Idle token
Start

RCV Ncs

 Idle

ε / token

τ

Ncs

RCV

Laboratory for Software Design, Iowa State University

Automatic Cut-off Generation for Parameterized Systems
Behavior of One Process in Any Environment

- **SND**
  - Start \(\rightarrow\) Idle \(\rightarrow\) token

- **RCV**
  - Idle \(\rightarrow\) Ncs \(\rightarrow\) choose

- **PASS**
  - Ncs \(\rightarrow\) Idle \(\rightarrow\) token

- **ENTER**
  - Ncs \(\rightarrow\) Cs \(\rightarrow\) in

\[\text{Laboratory for Software Design, Iowa State University}\]
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.

\[
sys(k) \models \varphi \iff \forall n > k : sys(n) \models \varphi
\]
System with 2 processes sys(2)

Topology = \{(token, i, (i + 1) \mod k),
\{(in, i, i),
(choose, i, 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 $k = 2$
- $\text{sys}(2) \models \varphi \iff \forall n > 2 : \text{sys}(2) \models \varphi$
Kind of Properties $\varphi$

Properties involving one process $i$

$\forall i, 1 \leq i \leq k : \text{sys}(k) \models \varphi(i) \iff \forall n \geq k, \forall i, 1 \leq i \leq n : \text{sys}(n) \models \varphi(i)$

Properties involving 2 interdependent processes $i, j$

$\forall i, j, 1 \leq i, j \leq k, i \neq j : \text{sys}(k) \models \varphi(i, j) \iff \forall n \geq k, \forall i, j, 1 \leq i, j \leq n, i \neq j : \text{sys}(n) \models \varphi(i, j)$
## Comparison with Other Techniques

<table>
<thead>
<tr>
<th>Topology</th>
<th>Existing Work</th>
<th>Our Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dining Philosophers</td>
<td>Ring</td>
<td>5^d i, j</td>
</tr>
<tr>
<td>Spin Lock</td>
<td>Star</td>
<td>3^e i, j</td>
</tr>
</tbody>
</table>


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)

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

Figure: Start state in sys(2)
System with 2 processes sys(2)

Figure: Autonomous action
System with 2 processes sys(2)

Figure: Non-autonomous Action

Topology = 
\{(\text{token}, i, i + 1), 
(\text{in}, i, i), 
(\text{choose}, i, i)\}
Generating a system
Proof of Soundness

System with 2 processes sys(2)

Topology = 
{(token, i, i + 1),
  (in, i, i),
  (choose, i, i)}
Proof of Soundness

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

Figure: $\text{sys}(n)$

Figure: $\text{sys}(k)$
Proof of Soundness

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

![Diagram of process transitions starting from a start state labeled 'i' and leading to another 'i'.]

**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: $sys(n)$