Algorithmic Analysis of Piecewise FIFO Systems

Naghmeh Ghafari
Richard Trefler
David R. Cheriton
School of Computer Science
University of Waterloo
Waterloo, ON, Canada

Arie Gurfinkel
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA

Nils Klarlund
Google
New York, NY
FIFO Systems

Definition

A **FIFO system** is a set of finite state machines that communicate over unbounded perfect FIFO channels.

A **common model of computation for distributed protocols:**

- IP-telecommunication protocols (BoxOS).
- Interacting web services (BPEL).
- System on Chip (SoC) architectures.

**Our Goal:**

Algorithmic analysis of safety properties in FIFO systems.
FIFO Systems in Action

Automaton $A_1$  Channels  Automaton $A_2$

Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle$
FIFO Systems in Action

Automaton $A_1$ | Channels | Automaton $A_2$

Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle$
FIFO Systems in Action

Automaton $A_1$  
Channels  
Automaton $A_2$

Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle \rightarrow \langle 2, 3, \varepsilon, d \rangle$
FIFO Systems in Action

Automaton $A_1$  Channels  Automaton $A_2$

Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle \rightarrow \langle 2, 3, \varepsilon, d \rangle \rightarrow \langle 2, 3, \varepsilon, dd \rangle \ldots$
An Alternative Execution

Automaton $A_1$ Channels Automaton $A_2$

Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle$
An Alternative Execution

Automaton $A_1$  Channels  Automaton $A_2$

Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle$
An Alternative Execution

Automaton $A_1$  

Channels

Automaton $A_2$

Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle \rightarrow \langle 2, 2, \varepsilon, b \rangle$
An Alternative Execution

Automaton $A_1$  Channels  Automaton $A_2$

Global Execution

\[ \langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle \rightarrow \langle 2, 2, \varepsilon, b \rangle \rightarrow \langle 3, 2, \varepsilon, \varepsilon \rangle \]

Error state reached!
From Reachability to Limit Languages

Inputs
Initial channel content: I
&
FIFO System

Reachable configurations partitioned by control location

L(A₁) : I    (¿d)* : I
From Reachability to Limit Languages

Inputs
Initial channel content: I
&
FIFO System

Reachable configurations partitioned by control location

L(A₁) : I  (?d)* : I
L(A₂) : I  (?d)*?a(?c!a)* : I
From Reachability to Limit Languages

Inputs
Initial channel content: I

FIFO System

Reachable configurations partitioned by control location

L(A_1) : I (\ (?d)^* : I
L(A_2) : I (\ (?d)^*?a(?c!a)^* : I
L(A_3) : I (\ (?d)^*?a(?c!a)^*!b(?c)^* : I
L(A_4) : I (\ (?d)^*?a(?c!a)^*?c : I
The Limit Language Problem

• Inputs
  – a language of actions: \( L \)
  – a set of initial channel contents: \( I \)

• Output
  – the set of all possible channel contents that result from zero or more application of \( L \) to \( I \).

This problem is undecidable in general. We focus on a particular class of systems for which it is decidable.
Motivation: BoxOS Protocol

The next generation telecommunication services over IP developed at AT&T Research.

feature 1  caller  callee  feature 2  caller  callee  feature 3
Motivation: BoxOS Protocol
Motivation: BoxOS Protocol

Setup Phase

Transparent Phase

Tear Down Phase
Piecewise Languages

• A language is **simply piecewise** if it can be expressed by a regular expression of the form:

\[ M_0^*a_0M_1^*\ldots a_{n-1}M_n^* \]

where \( M_i \subseteq \Sigma \) and \( a_i \in \Sigma \cup \{\varepsilon\} \)

• A language is **piecewise** if it is a finite (possibly empty) union of simply piecewise languages.

\[(a+b)^*c\] is simply piecewise where \( M_0 = \{a,b\} \) and \( a_0 = c \),

\[a^*c + b^*d\] is piecewise,

\[(ab)^*\] is NOT piecewise.

• A **partially ordered automaton** is a tuple \((A, \leq)\), where

  – \( A = (\Sigma, Q, q_0, \delta, F) \) is an automaton
  – \( \leq \subseteq Q \times Q \) is a partial order on states, \( q' \in \delta(q,a) \) implies that \( q \leq q' \).

**Theorem [Klarlund and Trefler, 04]**

A language is piecewise iff it is recognized by a partially ordered automaton.
**Piecewise FIFO Systems**

**Definition**

A FIFO system is **piecewise** if there exists a partial order on its control locations.

**Example**

A Piecewise FIFO System

```
1?d → 2!a  
1?d → 1!b  
2?a  
1?c  
1!b  

1?c  

2  
3  
4
```

NOT A Piecewise FIFO System

```
1?d → 2!a  
1?d → 1!b  
1!a  
2?a  
1?c  
1!b  

1?c  

2  
3  
4
```

**Observation**

In piecewise FIFO systems, action languages corresponding to limit languages are Kleene closure of sets of actions.
Outline

• Introduction
• FIFO Systems
• Limit Languages
• Motivation

• Piecewise FIFO Systems
  – Single Channel Systems (see paper)
  – Multi-Channel Systems

• Related Work
• Summary
Multi-Channel Communication Graph

Definition

A communication graph of a set of actions $S$ over channels $C$ is a directed graph $(C,E)$ where $(i, j) \in E$ iff there are $a$ and $b$ in $\Sigma$ such that $i?a \rightarrow j!b$ is an action in $S$.

Example

Act = \{1?a \rightarrow 2!b, 2?b \rightarrow 3!d, 3?a \rightarrow 3!a, 3?d \rightarrow 1!a\}

Our analysis is based on the topology of the communication graph

star  tree  inverted tree  DAG
Star Topology

Key Idea
Star topology algorithm is driven by the content of the origin channel.

Example
origin channel: $M_1*a_1 \ M_2*a_2$

<table>
<thead>
<tr>
<th>iterations</th>
<th>reachable configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>$(M_1<em>a_1M_2</em>a_2, \ )$</td>
</tr>
<tr>
<td>2nd</td>
<td>$(M_2*a_2, \ )$</td>
</tr>
<tr>
<td>3rd</td>
<td>$(\epsilon, \ )$</td>
</tr>
</tbody>
</table>

Each iteration of the algorithm is done using two functions:

SATURATE and STEP
SATURATE

Inputs
– initial channel configuration, \( I \), with the origin channel of the form \( M^* \cdot Z \)
– a set of actions: \( \text{Act} \)

Output
– the set of states that are reachable by reading an arbitrary number of letters from the head of the origin channel.

Example
\[
\text{Act} = \{1?a \rightarrow 2!a, 1?a \rightarrow 3!a, 1?b \rightarrow 2!b, 1?c \rightarrow 3!c\}
\]
**STEP**

**Inputs**
- initial channel configuration, $I$, with the origin channel of the form $(a_0 + \ldots + a_n) \cdot Z$
- a set of actions, $Act$

**Output**
- all configurations that are reachable by reading exactly one letter from the origin channel.

**Example**

$Act = \{1?a \to 2!a, 1?a \to 3!a, 1?b \to 2!b, 1?c \to 3!c\}$

\[
\langle (b+c), a^*, a^* \rangle \xrightarrow{\text{STEP}} \langle \varepsilon, a^*b, a^* \rangle
\]

\[
\langle (b+c), a^*, a^* \rangle \xrightarrow{\text{STEP}} \langle \varepsilon, a^*, a^*c \rangle
\]
Complexity Analysis

Theorem

In the worst case, the running time of the algorithm for computing the limit language in a $k$-channel system with a star topology is $O(\max(k^h, h))$, where $h$ is the size of the automaton representing the origin channel.

Proof

The depth of the recursion of the algorithm is bounded by $h$.

Inside each call, SATURATE takes constant time and returns a single configuration.

STEP returns a set of configurations with size bounded by $k-1$.

The complexity of the algorithm is bounded by the number of internal nodes of a $(k-1)$-ary tree of height $h$. 
Tree Topology

Star algorithm is not applicable!
– assumes all reads come from a single channel.

Observations:

1. Tree topology can be partitioned into a set of star topologies.

2. The communication graph induces a partial order of dependencies on channels:
   \[ i \leq j \text{ if there exists a path from } i \text{ to } j \text{ in the graph.} \]

Act_1 = \{1? \rightarrow 2!, 1? \rightarrow 3!, 1? \rightarrow 4!\}
Act_2 = \{2? \rightarrow 5!, 2? \rightarrow 6!\}
Act_3 = \{3? \rightarrow 7!\}
From Tree to Star

**Theorem**
For every sequence of actions \( x \), there exists a sequence \( y \) s.t.
- \( y \) has the same actions as \( x \)
- all reads of \( y \) are ordered
- If \( (x : w) \neq \emptyset \) for some \( w \), \( (y : w) = (x : w) \)

**Example**
\[
x = 2?c \rightarrow 4!c \ 1?a \rightarrow 2!a \ 3?d \rightarrow 5!d \ 1?b \rightarrow 3!b
\]
\[
w = \langle ab, c, d, \varepsilon, \varepsilon \rangle
\]
\[
y = 1?a \rightarrow 2!a \ 1?b \rightarrow 3!b \ 2?c \rightarrow 4!c \ 3?d \rightarrow 5!d
\]
\[
x : w \quad \langle \varepsilon, a, b, c, d \rangle
\]
\[
y : w \quad \langle \varepsilon, a, b, c, d \rangle
\]
Computing Limit Language

Algorithm Steps

Step 1  Partition the actions such that each partition is a star.
Step 2  Order the partitions according to the partial order on channels.
Step 3  Apply the Star algorithm on each partition following the order.

Algorithm in Action

Act* : I
(Act₁ * Act₂ * Act₃*) : I
Complexity Analysis

Assumptions

- The communication graph is an $N$-ary tree with $M$ internal nodes.
- The initial content of all the channels except the root is empty.

Theorem

In the worst case, the running time of the algorithm for computing the limit language in a $k$-channel system with a tree topology is $O(\max(N^h \times M, h^M))$, where $h$ is the size of the automaton representing the root content.

Proof

Each invocation of the Star algorithm produces at most $\max(N^h, h)$ piecewise configurations, each of size at most $h$.

There are at most $M$ number of invocations to the Star algorithm.
Inverted Tree Topology

Tree algorithm is not applicable!
- a channel may depend on several independent channels

Example

\[ \text{Act} = \{1?a \rightarrow 3!a, 2?b \rightarrow 3!b\} \quad w = \langle aa, bb, \varepsilon \rangle \]

\[ \langle \varepsilon, \varepsilon, abab \rangle \notin ((1?a \rightarrow 3!a) \ast (2?b \rightarrow 3!b)^\ast) : w \]

\[ ((2?b \rightarrow 3!b)^\ast (1?a \rightarrow 3!a)^\ast) : w \]
Shadow Channels

Shadow channels replace the nodes (channels) that have an in-degree greater than or equal to 2.

Algorithm for computing the limit language

Step 1  Introduce shadow channels and turn the graph into a tree.
Step 2  Use Tree algorithm to calculate the limit.
Step 3  Merge the contents of the shadow channels.

Example

Act = \{1?a \to 3!a, 2?b \to 3!b\} \quad w = \langle aa, bb, \varepsilon \rangle
\hat{\text{Act}} = \{1?\hat{a} \to \hat{3}_1!a, 2?\hat{b} \to \hat{3}_2!b\} \quad \hat{w} = \langle aa, bb, \varepsilon, \varepsilon \rangle
\hat{\text{Act}}: \hat{w} \xrightarrow{\text{Tree algorithm}} \langle \varepsilon, \varepsilon, aa, bb \rangle \quad \ldots \quad \hat{w} \xrightarrow{\text{Merge shadows}} \langle \varepsilon, \varepsilon, aa \parallel bb \rangle = \{\langle \varepsilon, \varepsilon, aabb \rangle, \langle \varepsilon, \varepsilon, bbaa \rangle, \langle \varepsilon, \varepsilon, baba \rangle, \langle \varepsilon, \varepsilon, abab \rangle, \langle \varepsilon, \varepsilon, baab \rangle, \langle \varepsilon, \varepsilon, abba \rangle\}
DAG Topology

Inverted Tree algorithm is not applicable!
- immediate predecessors of a channel may be interdependent.

Example

\[ \text{Act} = \{1?a \rightarrow 3!a, 1?b \rightarrow 2!b, 2?b \rightarrow 3!b\} \quad w = \langle a*b*, \varepsilon, \varepsilon \rangle \]

\[ \hat{\text{Act}} = \{1?a \rightarrow 3_1!a, 1?b \rightarrow 2!b, 2?b \rightarrow 3_2!b\} \quad \hat{w} = \langle a*b*, \varepsilon, \varepsilon, \varepsilon \rangle \]

\[ \hat{\text{Act}}^* : \hat{w} \quad \text{Tree algorithm} \Rightarrow \langle a*b*, \varepsilon, a*, \varepsilon \rangle \quad \text{Merge shadows} \Rightarrow \langle a*b*, \varepsilon, a* \rangle \quad \langle b, b, ba \rangle \]

includes infeasible configurations

Observation

While merging shadow channels, the dependencies between channels should be considered.
Indexed Merge

Modify merge to respect the dependencies between channels.

- Keep track of relative positions of each letter in a channel as it is copied between channels.
- Restrict the merge based on the history of positions of each letter.

Example

$$\text{Act} = \{1?a \rightarrow 3!a, 1?b \rightarrow 2!b, 2?b \rightarrow 3!b\} \quad w = \langle a^*b^*, \varepsilon, \varepsilon \rangle$$

$$\hat{\text{Act}} = \{1?a \rightarrow 3_1!a, 1?b \rightarrow 2!b, 2?b \rightarrow 3_2!b\}$$

$$\hat{w} = \langle a^*b^*, \varepsilon, \varepsilon, \varepsilon \rangle \quad \text{Add indices} \quad \hat{w}_{\text{idx}} = \langle a_1^*b_2^*, \varepsilon, \varepsilon, \varepsilon \rangle$$

$$\hat{\text{Act}}^* : \hat{w}_{\text{idx}} \quad \text{Tree algorithm} \quad \langle a_1^*b_2^*, \varepsilon, a_1^*, \varepsilon \rangle \quad \langle b_2^*, b_2^*, a_1^*, b_2^* \rangle $$

$$\text{Merge shadows} \quad \langle a^*b^*, \varepsilon, a^* \rangle \quad \langle b^*, b^*, a^*b^* \rangle$$
Boigelot et al. [SAS’97]
- QDDs to represent channel contents.
- Automata-theoretic algorithms to compute limit languages restricted to a single read, write, or conditional action.
- Semi-algorithms to compute sets of reachable states.

We consider limit languages of subsets of read, write, and conditional actions.

Klarlund and Trefler [AVOCS’04]
- Decidability and recognizability results for piecewise FIFO systems.

For single channel systems, our new algorithm is simpler.
For multi-channel systems, we give the first explicit algorithms.
## In Summary

Reachability problem in piecewise FIFO systems

<table>
<thead>
<tr>
<th>Initial Channel Language</th>
<th>Single Channel</th>
<th>Multi-channel with Acyclic CG</th>
<th>Multi-channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>piecewise</td>
<td>limit language</td>
<td>limit language</td>
<td>piecewise</td>
</tr>
<tr>
<td>regular</td>
<td>eff. piecewise</td>
<td>eff. piecewise</td>
<td>non-regular</td>
</tr>
<tr>
<td></td>
<td>exponential</td>
<td>non-regular [KT’04]</td>
<td>non-regular</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[KT’04]</td>
</tr>
</tbody>
</table>

References: [KT’04]
Questions?

THANK YOU FOR YOUR ATTENTION!