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:

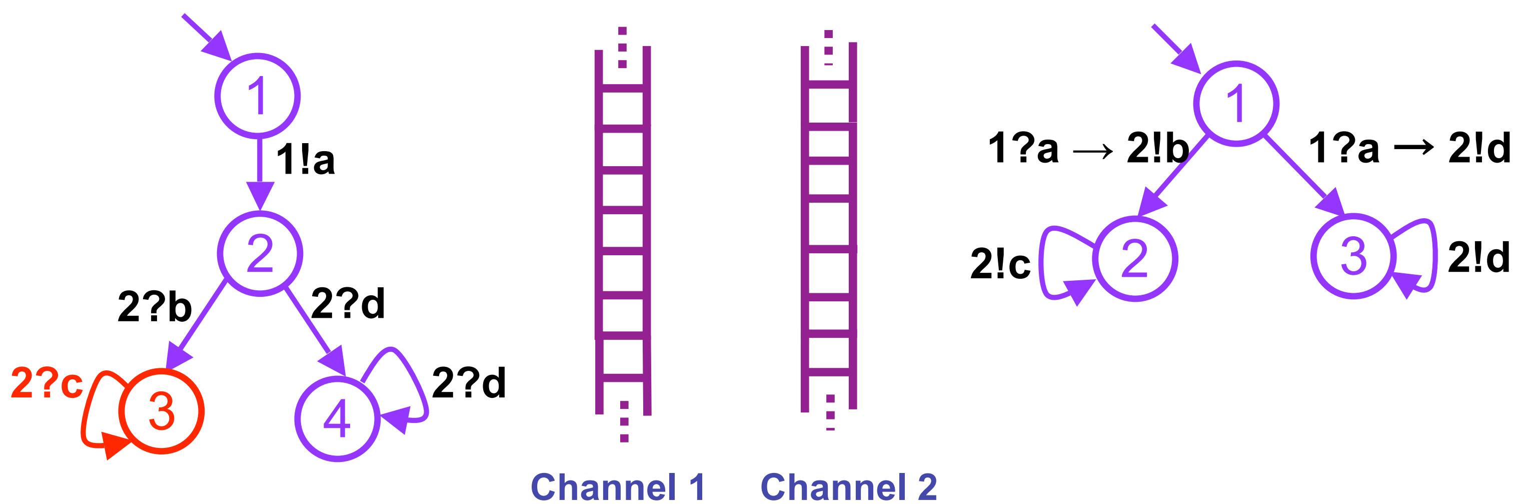
- IP-telecommunication protocols (BoxOS).
- interacting web services (BPEL).
- System on Chip (SoC) architectures.

Our Goal:

Algorithmic analysis of safety properties in FIFO systems.

FIFO Systems in Action

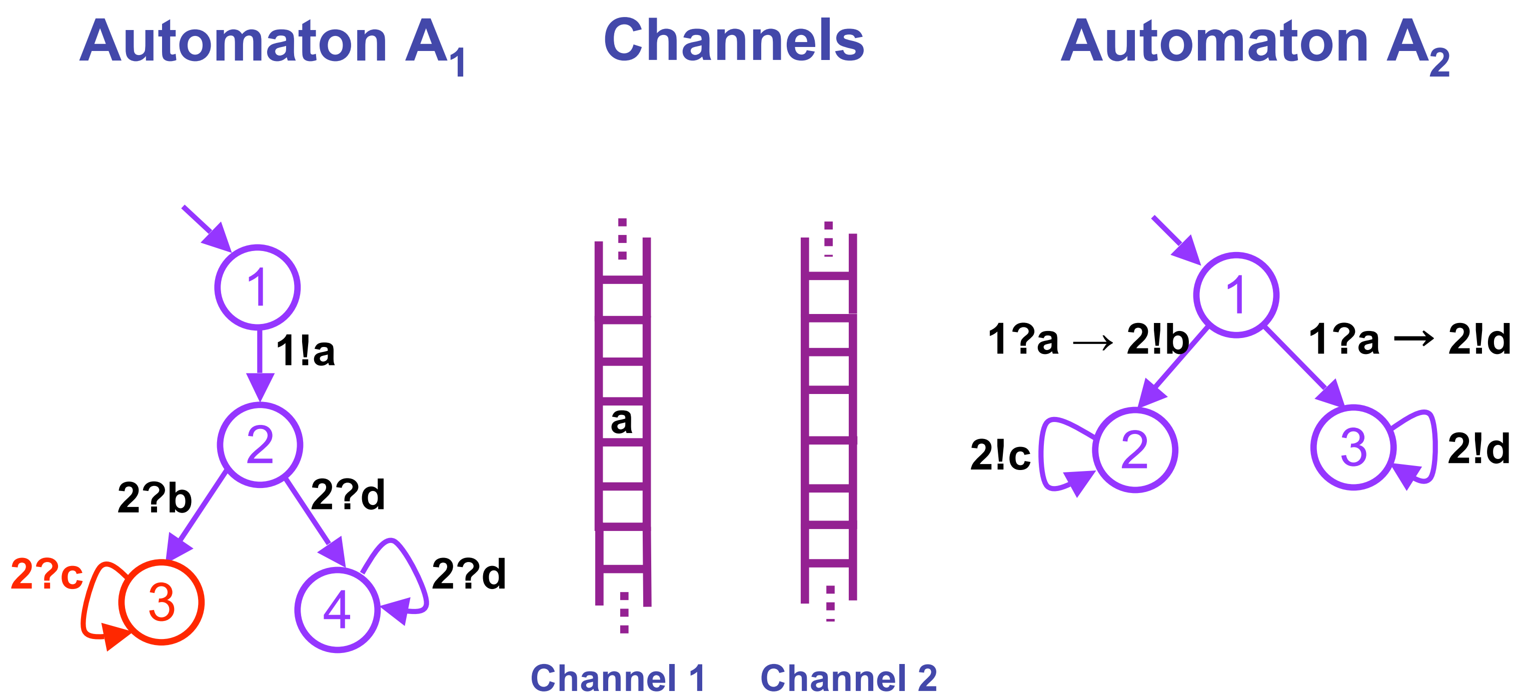
Automaton A_1 **Channels** **Automaton A_2**



Global Execution

⟨**1, 1,** ε**,** ε⟩

FIFO Systems in Action

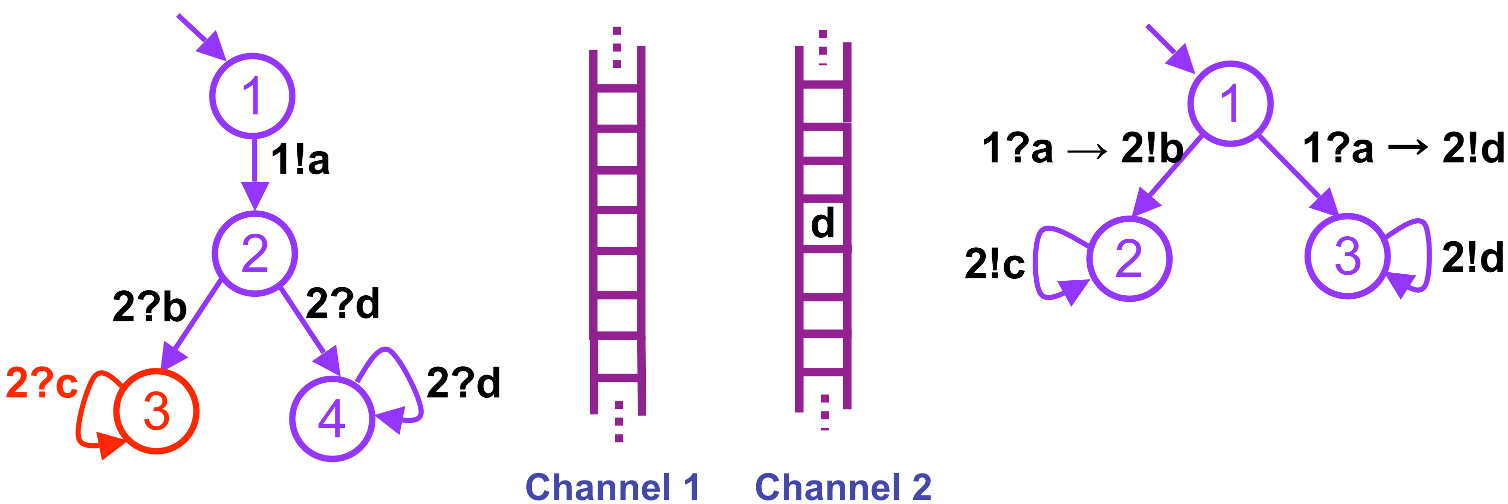


Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle$

FIFO Systems in Action

Automaton A_1 **Channels** **Automaton A_2**

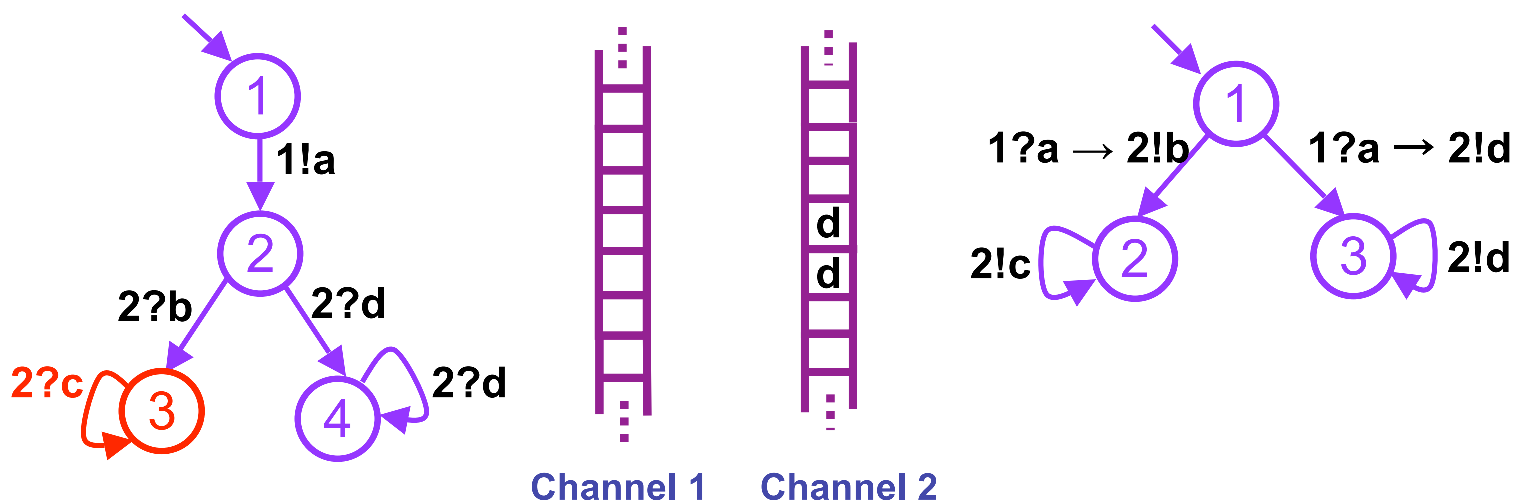


Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle \rightarrow \langle 2, 3, \varepsilon, d \rangle$

FIFO Systems in Action

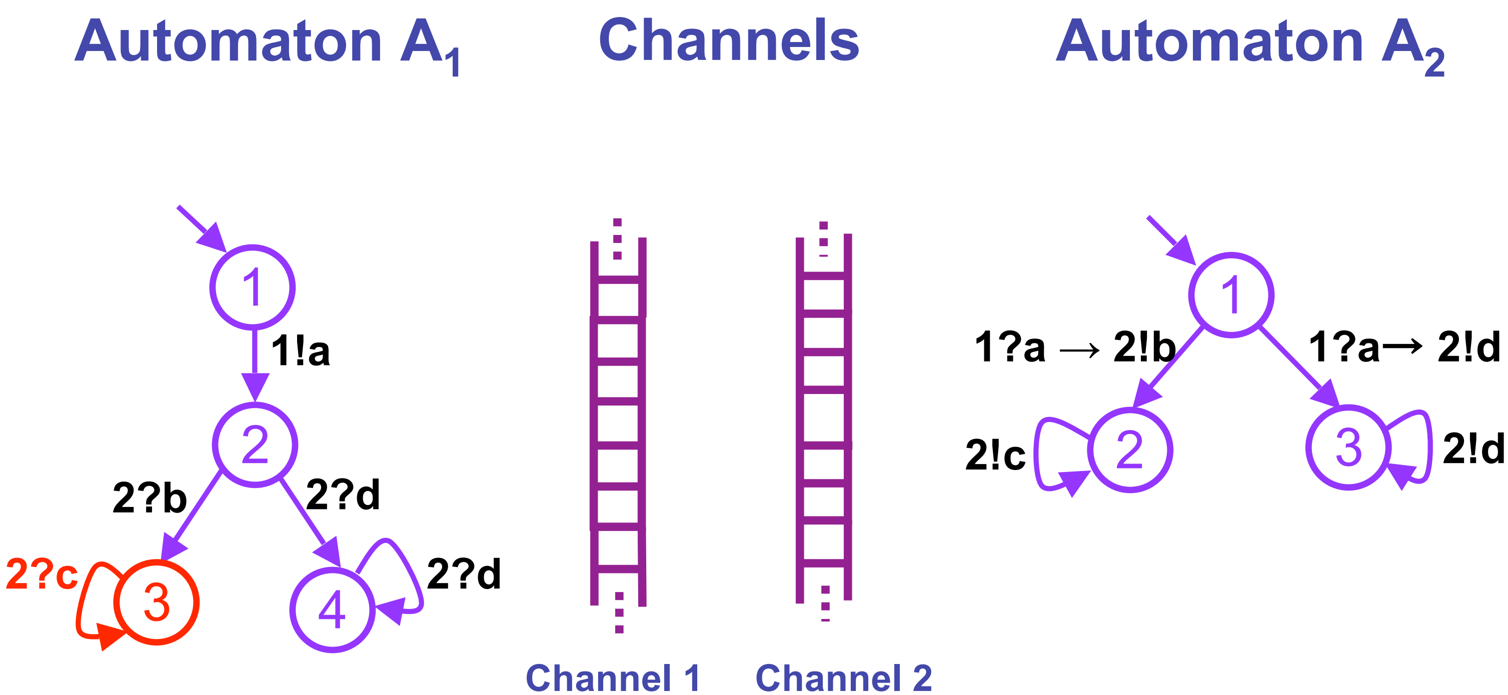
Automaton A_1 **Channels** **Automaton A_2**



Global Execution

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

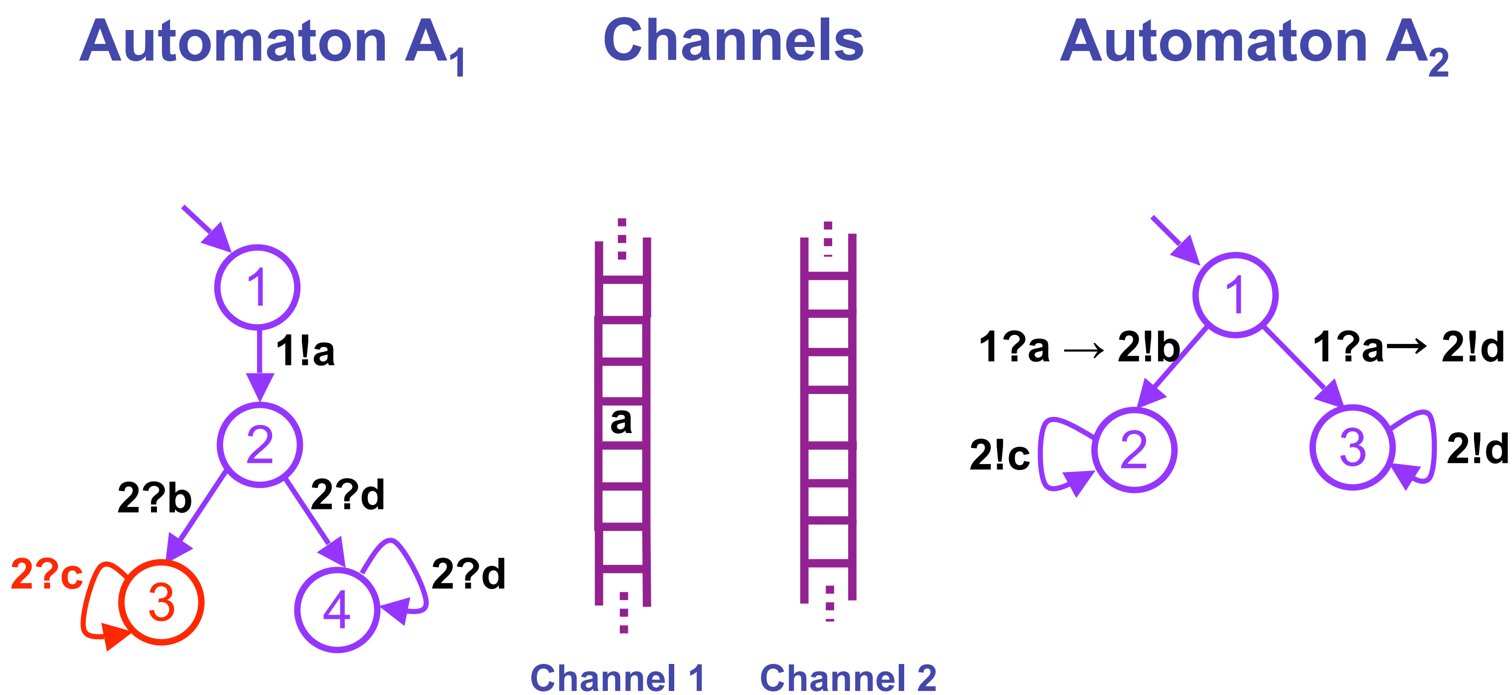
An Alternative Execution



Global Execution

⟨1, 1, ε, ε⟩

An Alternative Execution

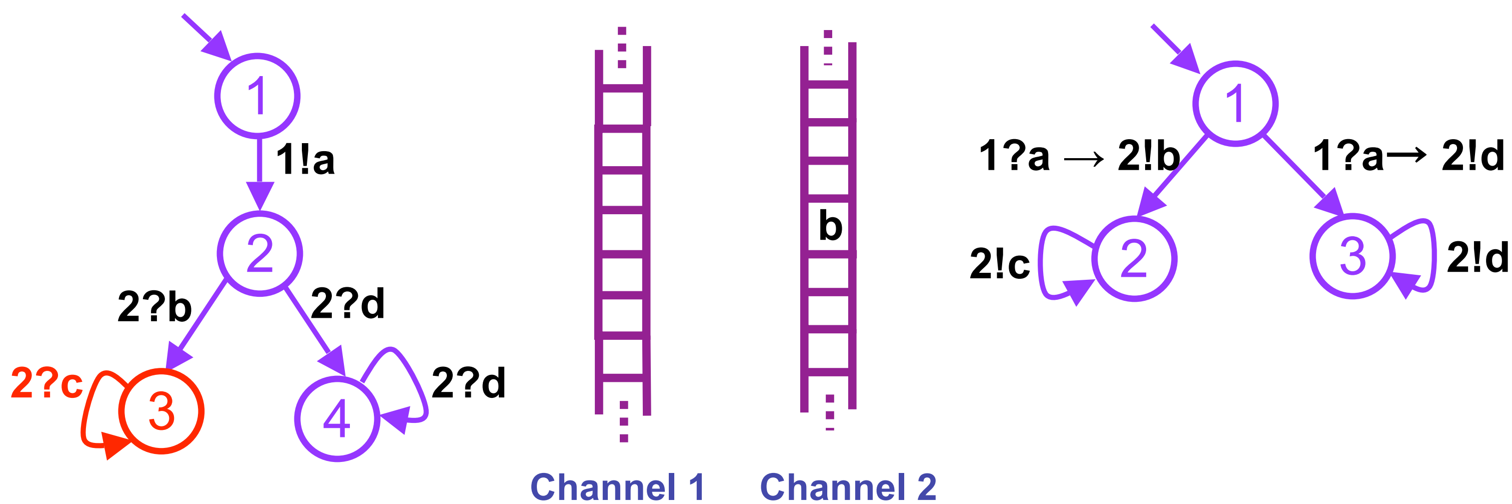


Global Execution

$\langle 1, 1, \varepsilon, \varepsilon \rangle \rightarrow \langle 2, 1, a, \varepsilon \rangle$

An Alternative Execution

Automaton A_1 | **Channels** | **Automaton A_2**

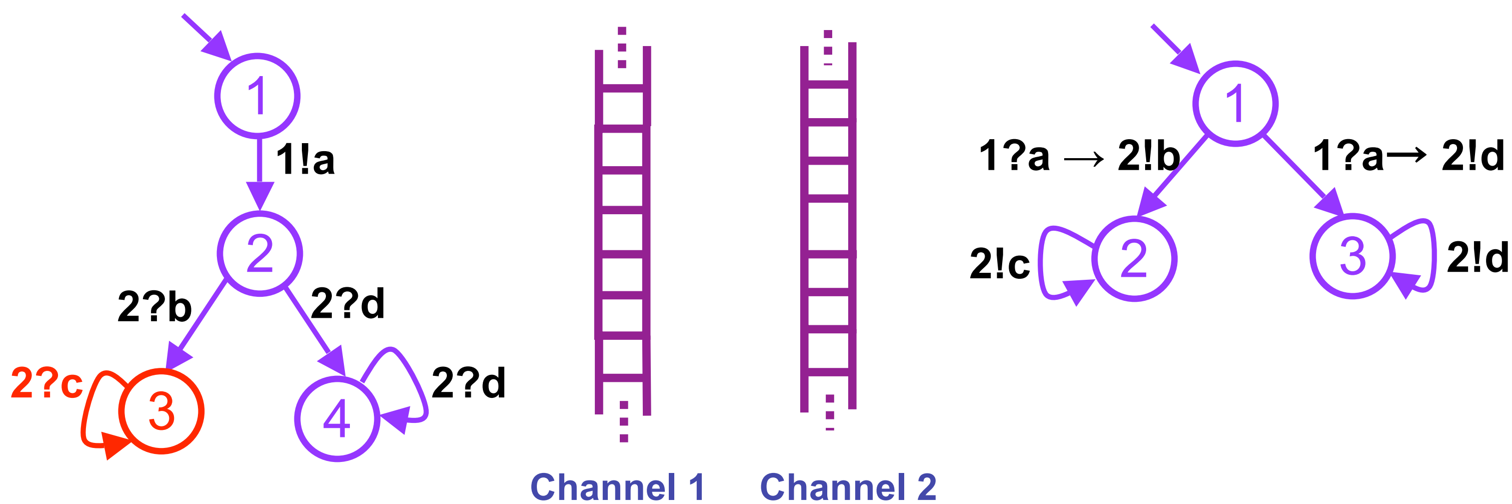


Global Execution

⟨1, 1, ε, ε⟩ → ⟨2, 1, a, ε⟩ → ⟨2, 2, ε, b⟩

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!

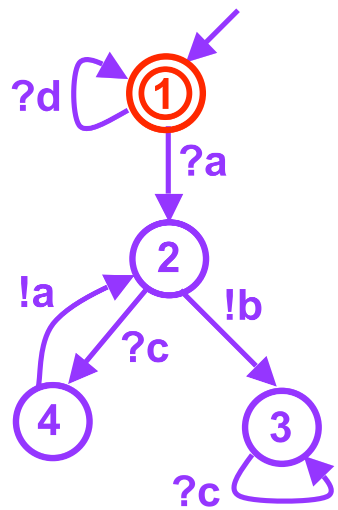
From Reachability to Limit Languages

Inputs

Initial channel content: **I**

&

FIFO System



Reachable configurations partitioned by control location

L(A_1) : I **(?d)* : I**

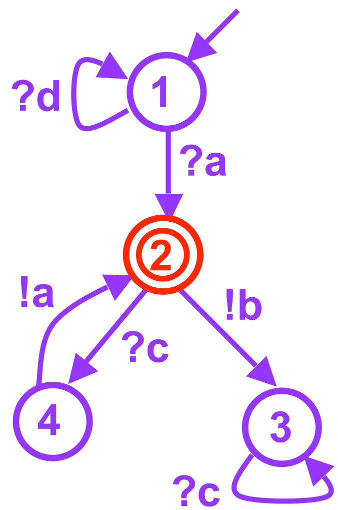
From Reachability to Limit Languages

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**

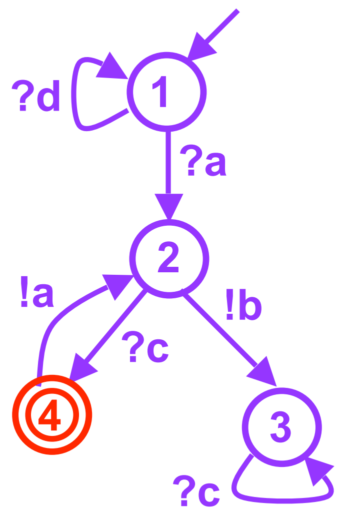
From Reachability to Limit Languages

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**.

Notation

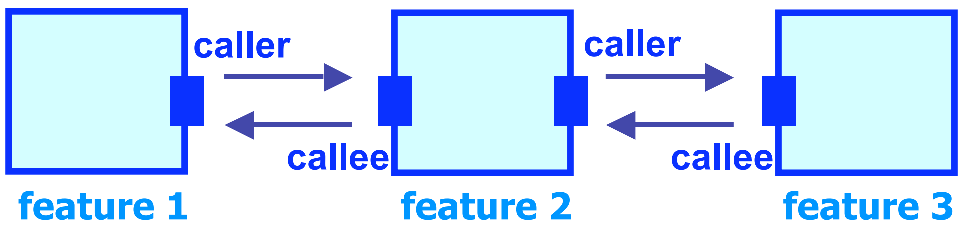
L* : I

This problem is undecidable in general.

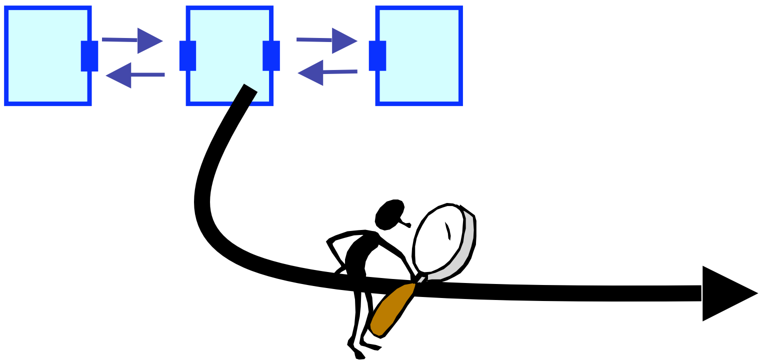
We focus on a particular class of systems for which it is decidable.

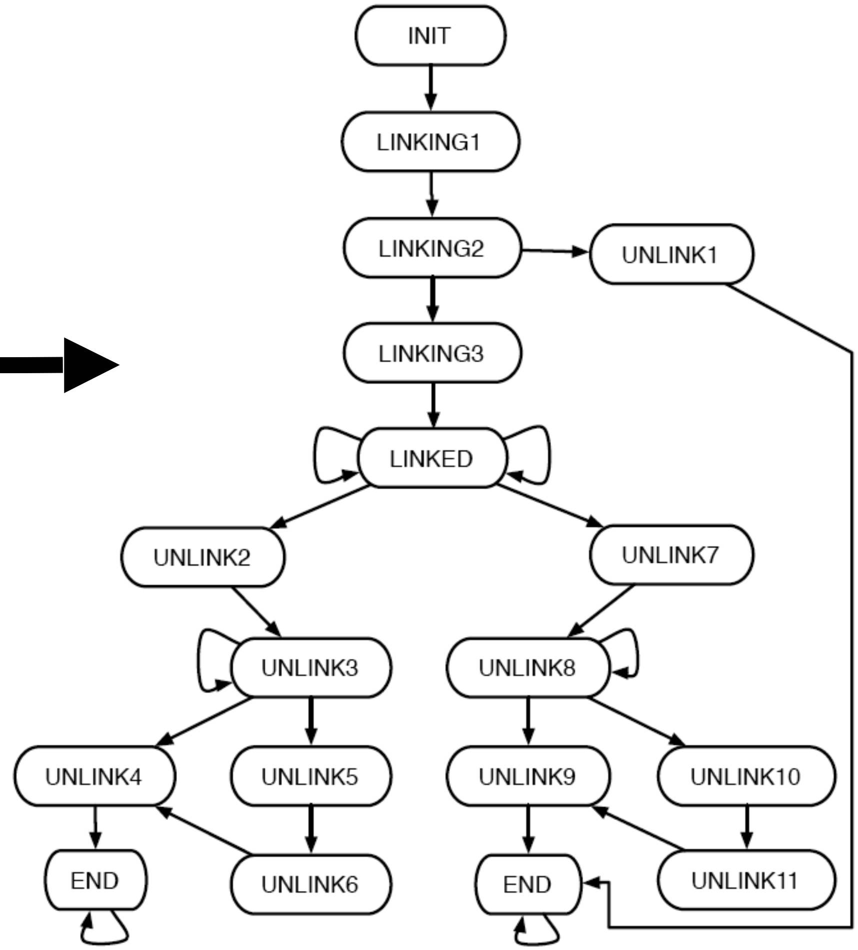
Motivation: BoxOS Protocol

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



Motivation: BoxOS Protocol



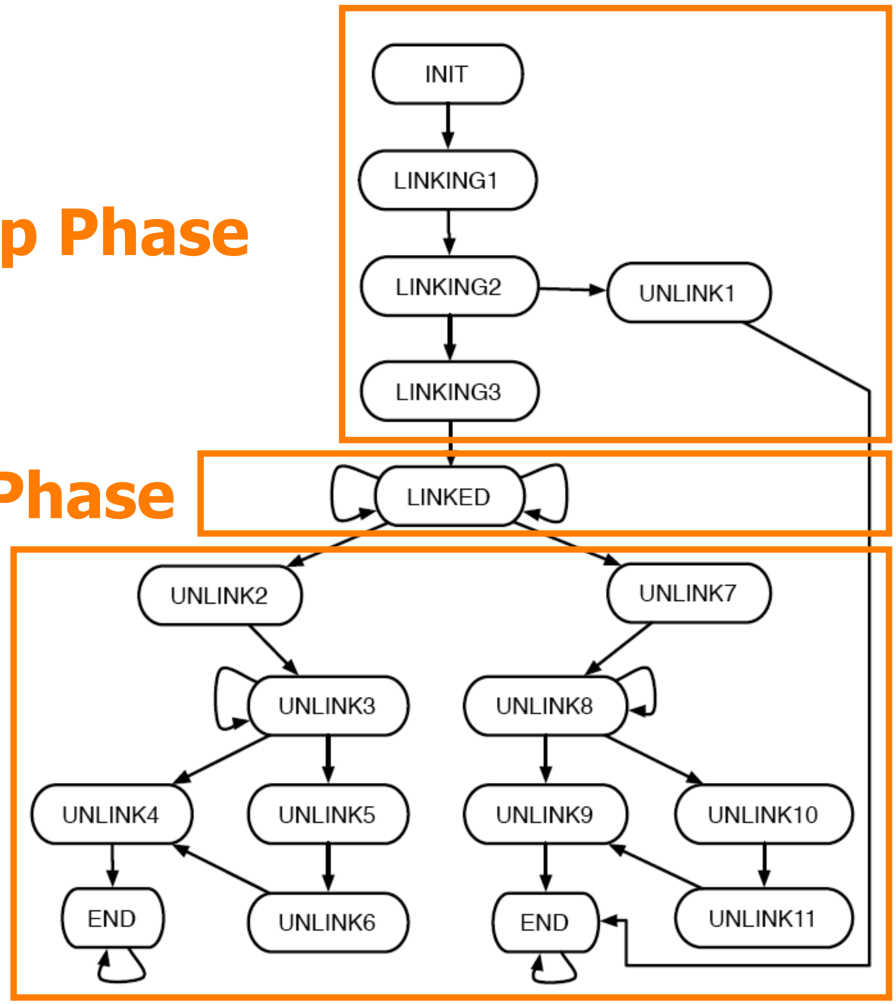


Motivation: BoxOS Protocol

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:**

  $M_0{}^*a_0\ M_1{}^*\ldots\ a_{n-1}M_n{}^*$ 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 , ≤), 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.

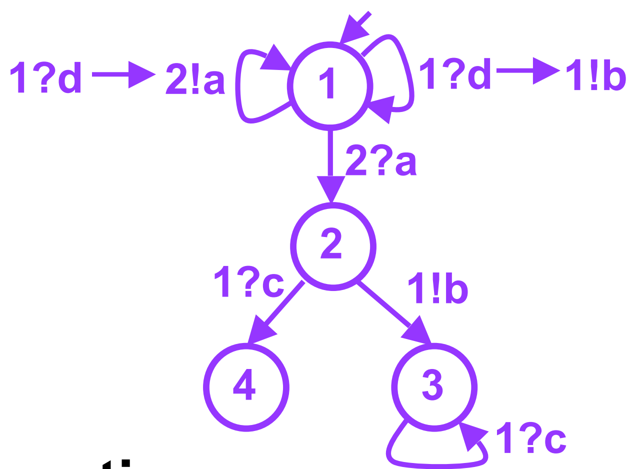
Piecewise FIFO Systems

Definition

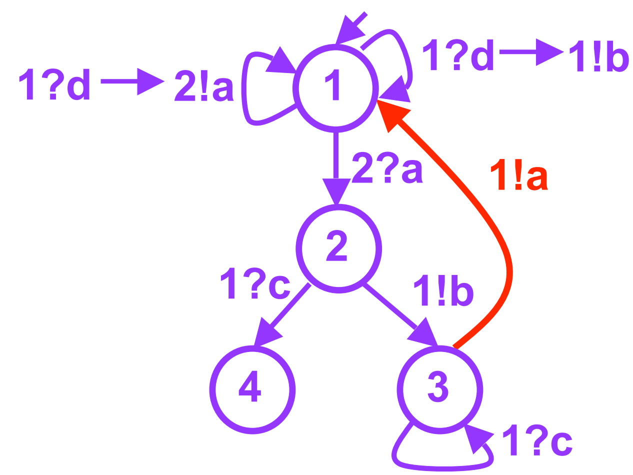
A FIFO system is piecewise if there exists a partial order on its control locations.

Example

A Piecewise FIFO System



NOT A Piecewise FIFO System



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**

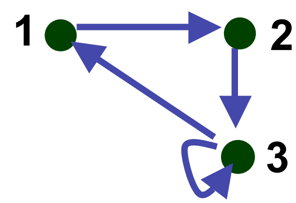
Multi-Channel Communication Graph

Definition

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

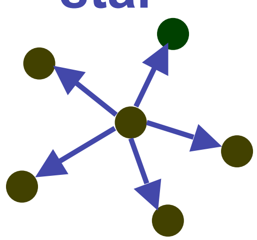
Example

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

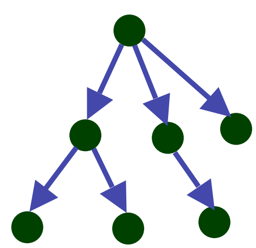


Our analysis is based on the topology of the communication graph

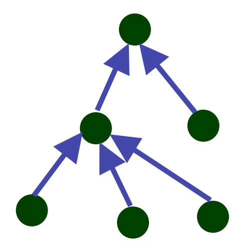
star



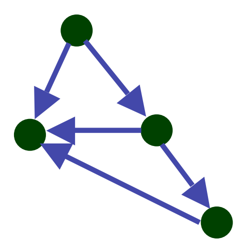
tree



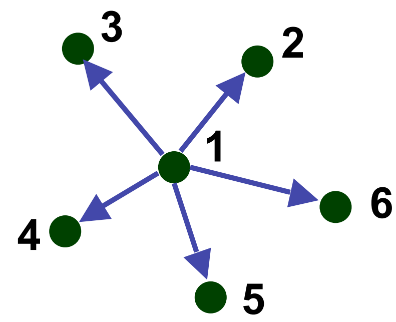
inverted tree



DAG



Star Topology



Key Idea

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

Example

origin channel: $M_1{}^*a_1\ M_2{}^*a_2$

iterations	reachable configurations
1st	$(M_1{}^*a_1 M_2{}^*a_2,\ \ )$
2nd	$(M_2{}^*a_2,\ \ )$
3rd	$(\varepsilon\ ,\ \ )$

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*· Z**
- a set of actions: **Act**

Output

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

Example

Act = {1?a → 2!a, 1?a → 3!a, 1?b → 2!b, 1?c → 3!c}

⟨a*(b+c), ε, ε⟩ **⟨a*(b+c), a*, a*⟩**

STEP

Inputs

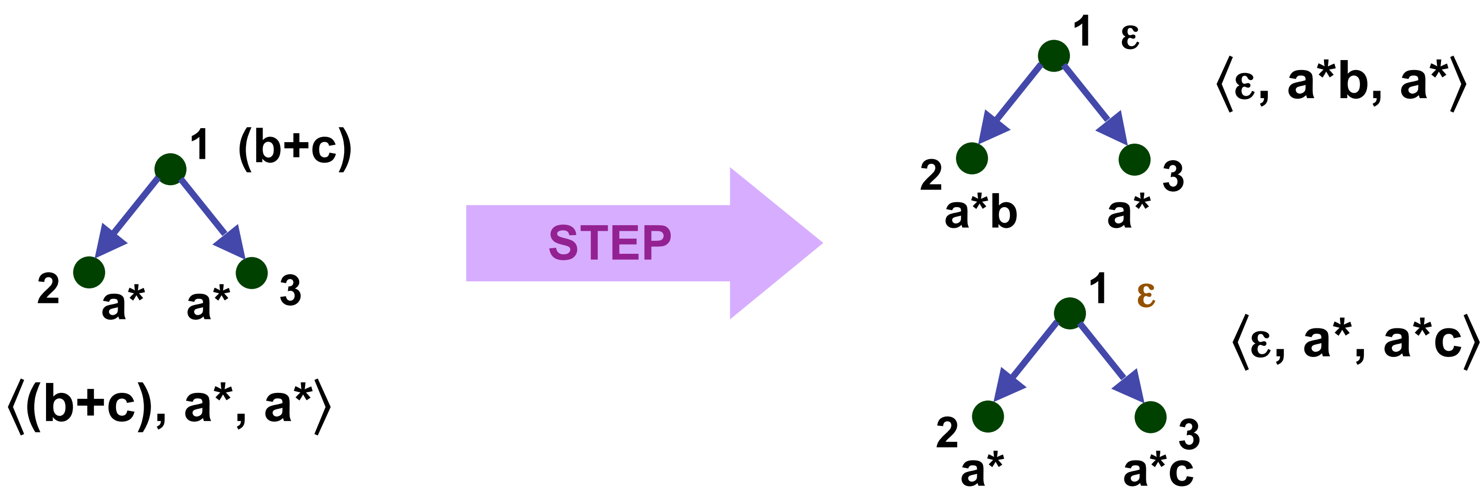
- initial channel configuration, **I**, with the origin channel of the form **$(a_0 + ... + a_n) \cdot Z$**
- a set of actions, **Act**

Output

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

Example

Act = {1?a → 2!a, 1?a → 3!a, 1?b → 2!b, 1?c → 3!c}



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

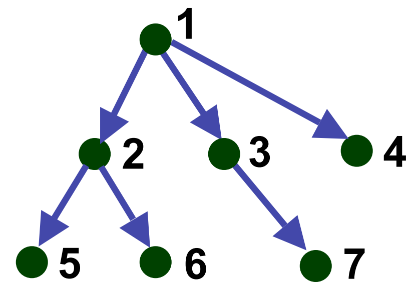
The depth of the recursion of the algorithm is bounded by h.

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:

$Act_1 = \{1? \to 2!, 1? \to 3!, 1? \to 4!\}$

$Act_2 = \{2? \to 5!, 2? \to 6!\}$

$Act_3 = \{3? \to 7!\}$

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

2. The communication graph induces a partial order of dependencies on channels:

   $i \leq j$ if there exists a path from i to j in the graph.

From Tree to Star

Theorem

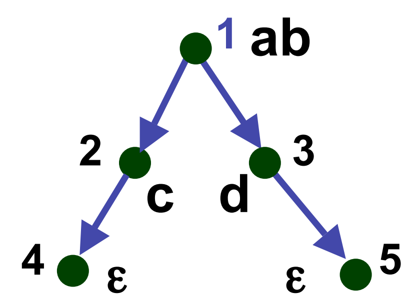
For every sequence of actions x, there exists a sequence y s.t.

- y has the same actions as x
- all reads of y are ordered
- If (x : w) ≠ Ø for some w, (y : w) = (x : w)

Example

x = 2?c → 4!c 1?a → 2!a 3?d → 5!d 1?b → 3!b

w = ⟨ab, c, d, ε, ε⟩



y = 1?a → 2!a 1?b → 3!b 2?c → 4!c 3?d → 5!d

x : w ⟨ε, a, b, c, d⟩

y : w ⟨ε, a, b, c, d⟩

Computing Limit Language

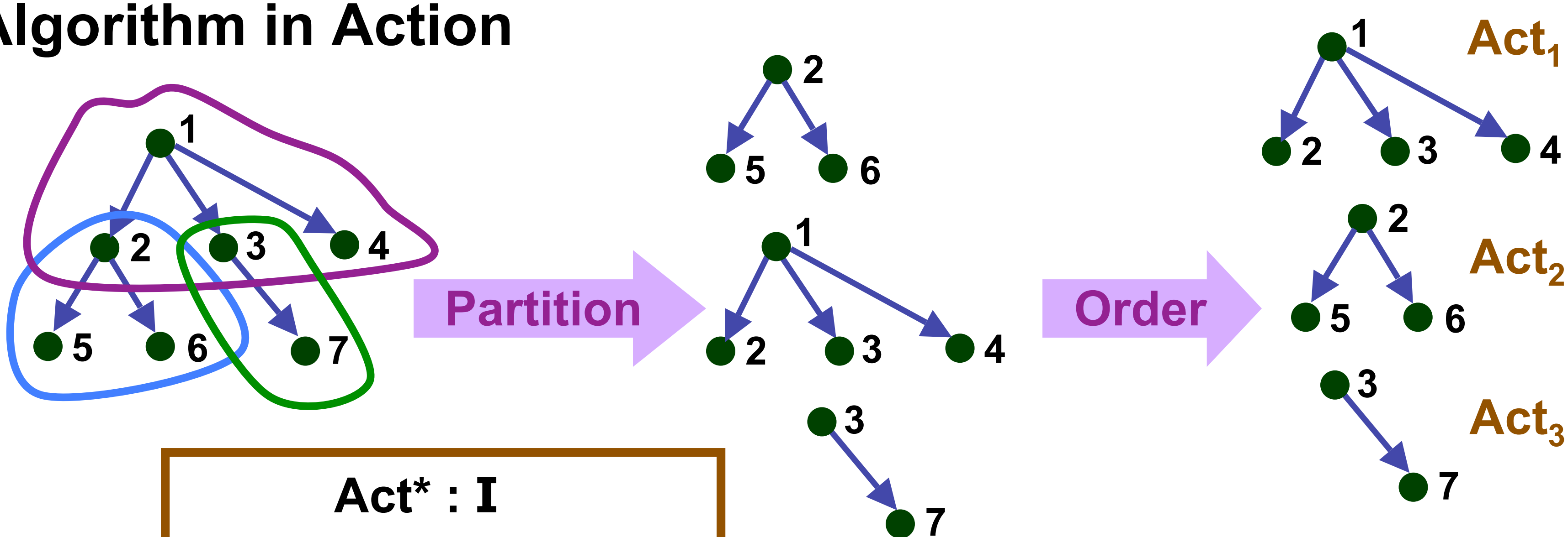
Algorithm Steps

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

(Act_1* Act_2* Act_3*) : 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

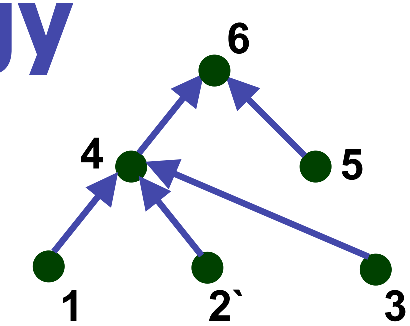
In the worst case, the running time of the algorithm for computing the limit language in a k-channel system with a tree topology is $O(\max(N^{h \times M}, h^M))$, where h is the size of the automaton representing the root content.

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

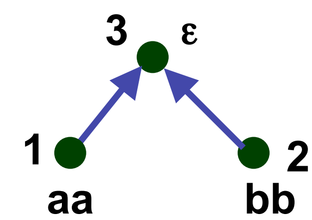


Tree algorithm is not applicable!

- a channel may depend on several independent channels

Example

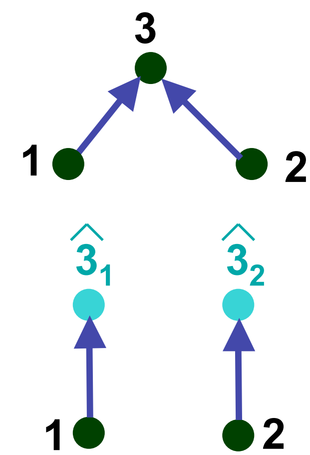
Act = {1?a ⟶ 3!a, 2?b ⟶ 3!b} **w = ⟨aa, bb, ε⟩**



⟨ε, ε, abab⟩ ∉
((1?a ⟶ 3!a)* (2?b ⟶ 3!b)*) : w
((2?b ⟶ 3!b)* (1?a ⟶ 3!a)*) : 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

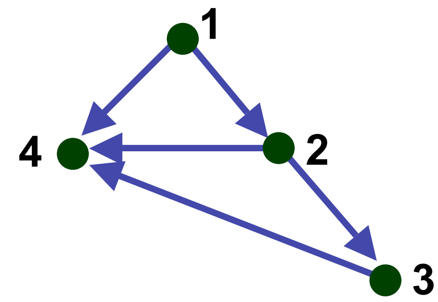
$Act = \{1?a \rightarrow 3!a,\ 2?b \rightarrow 3!b\}$ $\quad w = \langle aa, bb, \varepsilon \rangle$

$\widehat{Act} = \{1?a \rightarrow \hat{3}_1!a,\ 2?b \rightarrow \hat{3}_2!b\}$ $\quad \hat{w} = \langle aa, bb, \varepsilon, \varepsilon \rangle$

$\widehat{Act}^* : \hat{w}$ → **Tree algorithm** → … $\langle \varepsilon, \varepsilon, aa, bb \rangle$ … → **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

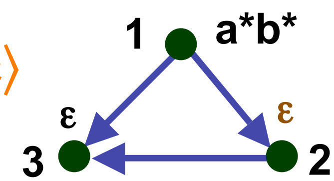


Inverted Tree algorithm is not applicable!

– immediate predecessors of a channel may be interdependent.

Example

Act = {1?a ⟶ 3!a, 1?b ⟶ 2!b, 2?b ⟶ 3!b} w = ⟨a*b*, ε, ε⟩



$\widehat{Act}$ = {1?a ⟶ $\hat{3}_1$!a, 1?b ⟶ 2!b, 2?b ⟶ $\hat{3}_2$!b} $\hat{w}$ = ⟨a*b*, ε, ε, ε⟩

$\widehat{Act}^*$: $\hat{w}$ ⟶ **Tree algorithm** ⟶ ⟨a*b*, ε, a*, ε⟩, ⟨b*, b*, a*, b*⟩ ⟶ **Merge shadows** ⟶ ⟨a*b*, ε, a*⟩, ⟨b*, b*, (a + b)*⟩

⟨b, b, ba⟩ includes infeasible configurations

Observation

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

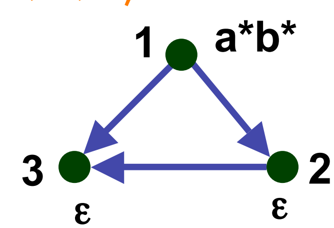
Indexed Merge

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

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



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

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

$\widehat{Act}^* : \widehat{w}_{idx}$ **Tree algorithm** → $\langle a_1{}^*b_2{}^*, \varepsilon, a_1{}^*, \varepsilon\rangle$, $\langle b_2{}^*, b_2{}^*, a_1{}^*, b_2{}^*\rangle$ **Merge shadows** → $\langle a^*b^*, \varepsilon, a^*\rangle$, $\langle b^*, b^*, a^*b^*\rangle$

(Most) Related Work

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

Initial Channel Language	Single Channel		Multi-channel with Acyclic CG		Multi-channel
	limit language	complexity	limit language	complexity	
piecewise	eff. piecewise	exponential	eff. piecewise	Star & Tree exponential	piecewise [KT’04]
regular	eff. regular		non-regular [KT’04]		non-regular [KT’04]

Questions?

THANK YOU FOR YOUR ATTENTION!