Bounded Reachability Problems are Decidable in FIFO Machines

Benedikt Bollig  Alain Finkel  Amrita Suresh

LSV – CNRS & ENS Paris-Saclay, Université Paris-Saclay, France

June 24, 2020
FIFO Machines

Distributed processes such that

- each process is a finite state machine
FIFO Machines

Distributed processes such that

- each process is a finite state machine
- there are a fixed number of processes

\[ P_i : \]

\[ P_1, \ldots, P_n \]
FIFO Machines

Distributed processes such that

- each process is a finite state machine
- there are a fixed number of processes
- they communicate using queues
FIFO Machines

- Studied since the 1980s. Widely used in distributed settings.
FIFO Machines

- Studied since the 1980s. Widely used in distributed settings.
- General model is difficult, hence underapproximations.
FIFO Machines

- Studied since the 1980s. Widely used in distributed settings.
- General model is difficult, hence underapproximations.
- Letter-bounded FIFO machines. ¹

¹Gouda et al., On deadlock detection in systems of communicating finite state machines, 1987.
FIFO Machines

- Studied since the 1980s. Widely used in distributed settings.
- General model is difficult, hence underapproximations.
- Letter-bounded FIFO machines. ¹
- Flat FIFO systems. ² ³

² Esparza et al., *Perfect Model for Bounded Verification*, 2012
³ Finkel and Praveen, *Verification of Flat FIFO Systems*, 2019
FIFO Machines

- Studied since the 1980s. Widely used in distributed settings.
- General model is difficult, hence underapproximations.
- Letter-bounded FIFO machines. ¹
- Flat FIFO systems. ² ³
- (Input-)Bounded FIFO machines strictly contain these subclasses.

²Esparza et al., *Perfect Model for Bounded Verification*, 2012
³Finkel and Praveen, *Verification of Flat FIFO Systems*, 2019
Example (Connection-Deconnection Protocol) \(^4\)

Example (Connection-Deconnection Protocol) \(^4\)

Initial configuration \((0, 0; \varepsilon, \varepsilon)\)

Example (Connection-Deconnection Protocol) \(^4\)

Run - \((0, 0; \varepsilon, \varepsilon) \xrightarrow{!a} (1, 0; a, \varepsilon)\)

---

Example (Connection-Deconnection Protocol) \(^4\)

\[
\begin{array}{c}
\text{(0, 0; } \varepsilon, \varepsilon \text{)} & \xrightarrow{!a} & (1, 0; a, \varepsilon) & \xrightarrow{?a} & (1, 1; \varepsilon, \varepsilon) \\
\end{array}
\]

Example (Connection-Deconnection Protocol)  

\[(0, 0; \varepsilon, \varepsilon) \xrightarrow{!a} (1, 0; a, \varepsilon) \xrightarrow{?a} (1, 1; \varepsilon, \varepsilon) \xrightarrow{!e} (1, 0; \varepsilon, e)\]  

---

\[\text{Jéron, Testing for unboundedness of FIFO channels, 1991.}\]
Example (Connection-Deconnection Protocol) \(^4\)

\[(0, 0; \varepsilon, \varepsilon) \xrightarrow{!a} (1, 0; a, \varepsilon) \xrightarrow{?a} (1, 1; \varepsilon, \varepsilon) \xrightarrow{!e} (1, 0; \varepsilon, e) \xrightarrow{!b} (0, 0; b, e)\]

A FIFO machine is a tuple $M = (Q, Ch, \Sigma, T, q_0)$ where
A FIFO machine is a tuple $M = (Q, Ch, \Sigma, T, q_0)$ where

$Q$ is a finite set of control-states.

$Q = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$. 
A FIFO machine is a tuple \( M = (Q, Ch, \Sigma, T, q_0) \) where

\[
0, 0 \quad \xrightarrow{c_1!a} \quad 1, 0 \\
\quad \xleftarrow{c_2?e} \quad 0, 1 \\
1, 0 \quad \xrightarrow{c_1!b} \quad 0, 0 \\
\quad \xleftarrow{c_2?e} \quad 1, 1 \\
1, 0 \quad \xrightarrow{c_1!b} \quad 1, 1 \\
\quad \xleftarrow{c_2!e} \quad 0, 1 \\
0, 1 \quad \xrightarrow{c_2!e} \quad 0, 0 \\
\quad \xleftarrow{c_1?a} \quad 1, 0 \\
1, 1 \quad \xrightarrow{c_1?a} \quad 1, 0 \\
\quad \xleftarrow{c_2!e} \quad 0, 1
\]

\( Ch \) is the number of channels.\

\( Ch = \{c_1, c_2\} \).
A FIFO machine is a tuple $M = (Q, Ch, \Sigma, T, q_0)$ where

\[ \Sigma \text{ is the alphabet.} \]
\[ \Sigma = \{a, b, e\}. \]
A FIFO machine is a tuple $M = (Q, Ch, \Sigma, T, q_0)$ where $T \subseteq Q \times A_M \times Q$ is the transition relation.
A FIFO machine is a tuple $M = (Q, Ch, \Sigma, T, q_0)$ where

$T \subseteq Q \times A_M \times Q$ is the transition relation where

$A_M = \{c!a \mid a \in \Sigma \text{ and } c \in Ch\} \cup \{c?a \mid a \in \Sigma \text{ and } c \in Ch\}$
Formal model

A FIFO machine is a tuple $M = (Q, Ch, \Sigma, T, q_0)$ where $q_0$ is the initial state. $q_0 = (0, 0)$. 
A configuration is \((q, w)\) where \(q\) is the control-state and \(w\) is a tuple of the channel contents. The set of configurations is \(S_M\).
A configuration is \((q, w)\) where \(q\) is the control-state and \(w\) is a tuple of the channel contents. The set of configurations is \(S_M\).

- \(\text{Reach}_M = \{ s \in S_M \mid (q_0, \varepsilon) \xrightarrow{\sigma} s \text{ for some } \sigma \in A^*_M \}\).
A configuration is \((q, w)\) where \(q\) is the control-state and \(w\) is a tuple of the channel contents. The set of configurations is \(S_M\).

\[
\text{Reach}_M = \{ s \in S_M \mid (q_0, \varepsilon) \xrightarrow{\sigma} s \text{ for some } \sigma \in A_M^* \}.
\]

Theorem

Testing the reachability of a configuration in a general FIFO system is undecidable. \(^a\)

Configurations and Reachability

- \( \text{Reach}_M(\sigma) = \{ s \in S_M \mid (q_0, \varepsilon) \xrightarrow{\sigma} s \} \) where \( \sigma \in A^*_M \).
Configurations and Reachability

- $\text{Reach}_M(\sigma) = \{ s \in S_M \mid (q_0, \varepsilon) \xrightarrow{\sigma} s \}$ where $\sigma \in A^*_M$.
- $\text{Reach}_M(L) = \bigcup_{\sigma \in L} \text{Reach}_M(\sigma)$. 
We define the *send projection over c* \( \text{proj}_c! : A^*_M \rightarrow \Sigma^* \)

Example: \( \text{proj}_c!(c!x.d!y.c?x.c!z.c!z) = xzz \)
Let $w_1, ..., w_n \in \Sigma^+$ be non-empty words where $n \geq 1$. $L$ is a bounded language over $(w_1, ..., w_n)$ if $L \subseteq w_1^*...w_n^*$. 
Let $w_1, \ldots, w_n \in \Sigma^+$ be non-empty words where $n \geq 1$. $L$ is a bounded language over $(w_1, \ldots, w_n)$ if $L \subseteq w_1^* \ldots w_n^*$.

$(ab)^* d(c)^*$ is a bounded language over $(ab, d, c)$.
Let $w_1, \ldots, w_n \in \Sigma^+$ be non-empty words where $n \geq 1$. $L$ is a **bounded language** over $(w_1, \ldots, w_n)$ if $L \subseteq w_1^* \ldots w_n^*$.

$(ab)^*d(c)^*$ is a bounded language over $(ab, d, c)$.

$((ab)^*(cd)^*)^*$ is not a bounded language.
Let \( w_1, \ldots, w_n \in \Sigma^+ \) be non-empty words where \( n \geq 1 \).

\( L \) is a **bounded language** over \((w_1, \ldots, w_n)\) if \( L \subseteq w_1^* \ldots w_n^* \).

Let \( L = (L_c)_{c \in Ch} \) be non-empty regular bounded languages over \( \Sigma \).

\( L! = \{ w \in A_M^* \mid proj_c(w) \in L_c \text{ for all } c \in Ch \} \).
Let $w_1, \ldots, w_n \in \Sigma^+$ be non-empty words where $n \geq 1$. 
$L$ is a **bounded language** over $(w_1, \ldots, w_n)$ if $L \subseteq w_1^* \ldots w_n^*$.

Let $L = (L_c)_{c \in Ch}$ be non-empty regular bounded languages over $\Sigma$.
$L! = \{w \in \mathbb{A}^*_M \mid \text{proj}_c(w) \in L_c \text{ for all } c \in Ch\}$.
(We define $L?$ similarly.)
Input-Bounded Reachability Problem

Given

- a FIFO machine $M = (Q, Ch, \Sigma, T, q_0)$,
Input-Bounded Reachability Problem

Given

- a FIFO machine $M = (Q, Ch, \Sigma, T, q_0)$,
- a control-state $q \in Q$, 
Input-Bounded Reachability Problem

Given

- a FIFO machine $M = (Q, Ch, \Sigma, T, q_0)$,
- a control-state $q \in Q$,
- channel contents $w$, and
Input-Bounded Reachability Problem

Given

- a FIFO machine $M = (Q, Ch, \Sigma, T, q_0)$,
- a control-state $q \in Q$,
- channel contents $w$, and
- a tuple $L = (L_c)_{c \in Ch}$ of non-empty regular bounded languages over $\Sigma$. 

Amrita Suresh (LSV)
Input-Bounded Reachability Problem

Given

- a FIFO machine $M = (Q, Ch, \Sigma, T, q_0)$,
- a control-state $q \in Q$,
- channel contents $w$, and
- a tuple $L = (L_c)_{c \in Ch}$ of non-empty regular bounded languages over $\Sigma$

Question: Do we have $(q, w) \in \text{Reach}_M(L!)$?
Theorem

The Input-Bounded Reachability Problem is decidable.
Theorem

*The Input-Bounded Reachability Problem is decidable.*

Proof using counter machines...
A counter machine (with zero tests) is a tuple $C = (Q, Cnt, T, q_0)$ where

- $Q$ is the finite set of control-states. $Q = \{q_0, q_1, q_2\}$
A counter machine (with zero tests) is a tuple $C = (Q, Cnt, T, q_0)$ where

- $Q$ is a non-empty finite set of states.
- $Cnt$ is a non-empty finite set of counters. $Cnt = \{x, y\}$
- $T$ is a transition relation.
- $q_0$ is the initial state.
Counter machines

A counter machine (with zero tests) is a tuple \( C = (Q, Cnt, T, q_0) \) where

- \( q_0 \) is the initial state.

\[ q_0 \xrightarrow{x++} q_1 \xrightarrow{x--} q_2 \xrightarrow{y++} q_1 \]

\[ q_0 \xrightarrow{x++} q_2 \xrightarrow{x=0} q_0 \]
A counter machine (with zero tests) is a tuple $C = (Q, Cnt, T, q_0)$ where

- $T \subseteq Q \times A_C \times Q$ is the transition relation, where $A_C = \{x++, x-- \mid x \in Cnt\} \times 2^{Cnt}$. 

![Diagram of counter machine with states $q_0$, $q_1$, and $q_2$, transitions $x++$, $x--$, and $y++$.](image-url)
Counter machines with RESTRICTED zero tests

Once a counter has been tested for zero, it cannot be incremented or decremented any more.
Counter machines with RESTRICTED zero tests

Formally, let $L_{\text{zero}}$ be the set of words $(op_1(x_1), z_1) \ldots (op_n(x_n), z_n) \in A_C^*$. 
Counter machines with RESTRICTED zero tests

Formally, let $L_{zero}$ be the set of words $(op_1(x_1), Z_1) \ldots (op_n(x_n), Z_n) \in A_C^*$. Then, for every two positions $1 \leq i \leq j \leq n$, we have $x_j \notin Z_i$. 
Counter machines with RESTRICTED zero tests
Counter machines with RESTRICTED zero tests

Theorem

The following problem is decidable: Given a counter machine $C = (Q, Cnt, T, q_0)$, a regular language $L \subseteq A_C^*$, a control state $q \in Q$, and counter valuation $v$, do we have $(q, v) \in \text{Reach}_C(L_{\text{zero}} \cap L)$?
Intuition: Given a bounded language $L$, which is bounded over $(w_1, \ldots, w_n)$, we construct a counter $x_i$ for each $w_i$. 
\[ \hat{L}_c = (ab)^* bb^* \]
Step 1: Distinct letter property

\[ \hat{L}_c = (ab)^* bb^* \]

\[ L_c = (a_1a_2)^* a_3a_3^* \]
Step 2: Trace property

\[ L_c = (a_1 a_2)^* a_3 a_3^* \]
Step 2: Trace property

\[ A \text{ for } L! \cap \text{Pref}(L?) \]

\[ L_c = (a_1a_2)^*a_3a_3^* \]
Step 3: Normal form

\[ L_c = (a_1a_2)^*a_3a_3^* \]
Step 4: Conversion to counter machine

\[ L_c = (a_1 a_2)^* a_3 a_3^* \]
Equivalence between configurations

Given the normal form and counter automata, is there a 1-1 equivalence between the configurations?
Given the normal form and counter automata, is there a 1-1 equivalence between the configurations? 

NO!

Given a counter configuration $(q; 3, 0)$ for some $q$, where $L = (ab)^*(c)^*$, what is the corresponding FIFO machine configuration?
Equivalence between configurations

Given the normal form and counter automata, is there a 1-1 equivalence between the configurations?
But we can keep track of the last message sent.
Equivalence between configurations

Given the normal form and counter automata, is there a 1-1 equivalence between the configurations?

\( L_{a}^{\text{last}} \subseteq A_{M}^{*} \) be the set of words where \( a \) describes the last sent messages.
## Other bounded problems

### Table: Summary of key results. (D stands for Decidable)

|               | Letter-bounded | Bounded $|Ch| = 1$ | Bounded $|Ch| > 1$ |
|---------------|----------------|-----------|-------------|
| UNBOUND       | D              | D         | D $^5$      |
| TERM          | D              | **EXPTIME** | D          |
| REACH         | D $^6$         | **EXPTIME** | D, not ELEM |
| CS-REACH      | D              | **EXPTIME** | D          |

---


Future work

- Precise complexity for termination and boundedness
- Upper bounds for single channel case
- Output bounded reachability problems