Motivating Example: Sliding Window Protocol

Systems of Communicating One-Counter Machines and their Topology Parametrized Reachability Problem

Main Theorem: Proof of “Only If” Direction

Main Theorem: Proof of “If” Direction

Related/On-going/Future Work
Motivating Example: Sliding Window Protocol

Systems of Communicating One-Counter Machines and their Topology Parametrized Reachability Problem

Main Theorem: Proof of “Only If” Direction

Main Theorem: Proof of “If” Direction

Related/On-going/Future Work
Example Sliding Window Protocol

- One local counter for each process \((x, y)\)

- Asynchronous communication via perfect channels \((c, d)\)
  - send the counter's value
  - receive and test/overwrite the counter

- Sources of infinity: local counters, message alphabet, channel length
Example Sliding Window Protocol

- One local counter for each process \((x, y)\)
- Asynchronous communication via perfect channels \((c, d)\)
  - send the counter’s value
  - receive and test/overwrite the counter
Example Sliding Window Protocol

- One local counter for each process \((x, y)\)
- Asynchronous communication via perfect channels \((c, d)\)
  - send the counter’s value
  - receive and test/overwrite the counter
- Sources of **infinity**: local counters, message alphabet, channel length
Goal:
Check absence of unspecified receptions due to $y$ being too large
Goal:
Check absence of unspecified receptions due to $y$ being too large

\[
\begin{align*}
\text{send}(c, x) & \quad x++ \\
\text{x} & \quad := \text{recv}(d)
\end{align*}
\]

\[
\begin{align*}
y & \quad := \text{recv}(c) \\
\text{send}(d, y) & \quad y--
\end{align*}
\]
Safety Verification of Example

Goal:
Check absence of unspecified receptions due to $y$ being too large

More formal goal:
Check that $err$ is not reachable
An Erroneous Execution

```
An Erroneous Execution

send(c, x)  \rightarrow  x++  \rightarrow  x := recv(d)

send(c, x)  \rightarrow  x++  \rightarrow  x := recv(d)  \rightarrow  send(c, x)

y++  \rightarrow  y == recv(c)  \rightarrow  y--

send(d, y)  \rightarrow  y == recv(c)  \rightarrow  y++  \rightarrow  y == recv(c)

y == recv(c)  \rightarrow  y++  \rightarrow  send(d, y)  \rightarrow  y == recv(c)

err

0

1

1

0

1

1

y == recv(c)

y == recv(c)

y == recv(c)

y == recv(c)
```
Motivating Example: Sliding Window Protocol

Systems of Communicating One-Counter Machines and their Topology Parametrized Reachability Problem

Main Theorem: Proof of “Only If” Direction

Main Theorem: Proof of “If” Direction

Related/On-going/Future Work
A topology is $T = \langle P, C, \text{src}, \text{dst} \rangle$ where

- $P$ : finite set of processes
- $C$ : finite set of channels
- $\text{src}, \text{dst} : C \to P \times \{\bullet, \circ\}$

**Definition:**

Communication types

- **strong**: standard CFSM-style communication ($\Rightarrow$)
- **weak**: counter lost by communication ($\Rightarrow$)
Definition:

A communicating one-counter machine is \( \langle S, I, F, A, \Delta \rangle \) where

- \( S \) : finite set of states
- \( I, F \subseteq S \) : initial and final states
- \( A \) : finite set of actions
- \( \Delta \subseteq S \times A \times S \) : finite set of transition rules
A communicating one-counter machine is \( \langle S, I, F, A, \Delta \rangle \) where

- \( S \) : finite set of states
- \( I, F \subseteq S \) : initial and final states
- \( A \) : finite set of actions
- \( \Delta \subseteq S \times A \times S \) : finite set of transition rules

**Definition:**

**Actions:** \( \text{add}(k) \mid \text{test}(\varphi) \mid c! \mid c? \) \( (k \in \mathbb{Z}, \varphi \in \text{Presb}_{1}, c \in C) \)
A communicating one-counter machine is \( \langle S, I, F, A, \Delta \rangle \) where

\[ \Rightarrow S : \text{finite set of states} \]
\[ \Rightarrow I, F \subseteq S : \text{initial and final states} \]
\[ \Rightarrow A : \text{finite set of actions} \]
\[ \Rightarrow \Delta \subseteq S \times A \times S : \text{finite set of transition rules} \]

**Definition:**

A system of communicating one-counter machines is \( \langle T, (M^p)_{p \in P} \rangle \) where

\[ \Rightarrow T : \text{topology} \]
\[ \Rightarrow M^p : \text{communicating one-counter machine} \]
SC1CM Semantics: Configurations

A SC1CM is $\langle T, (M^p)_{p \in P} \rangle$ where $M^p = \langle S^p, I^p, F^p, A^p, \Delta^p \rangle$

A configuration is $\Pi_{p \in P} S^p \cup s \cup \mathbb{N}^p \cup x \cup (\mathbb{N}^*)^C \cup w$

$initial \iff s^p \in I^p \land x = 0 \land w = \varepsilon$

$final \iff s^p \in F^p$
A SC1CM is $\langle T, (M^p)_{p \in P} \rangle$ where $M^p = \langle S^p, I^p, F^p, A^p, \Delta^p \rangle$

Recall:

The transition relation $(s, x, w) \xrightarrow{a} (s', x', w')$ is defined by

- exactly one process moves
- counter actions behave as expected
Recall:

A SC1CM is $\langle T, (M^p)_{p \in P} \rangle$ where $M^p = \langle S^p, I^p, F^p, A^p, \Delta^p \rangle$

The transition relation $(s, x, w) \xrightarrow{a} (s', x', w')$ is defined by

- exactly one process moves
- counter actions behave as expected
- communication actions depend on the endpoint’s type

$\bullet \xrightarrow{c} c! \equiv c!x$

$\circ \xrightarrow{c} c! \equiv c!x ; x := \text{any}$

$\bullet \xleftarrow{c} c? \equiv c?x$

$\circ \leftarrow{c} c? \equiv x := \text{any} ; c?x$
SC1CM Semantics: Transitions

A SC1CM is $\langle \mathcal{T}, \{M^p\}_{p \in P}\rangle$ where $M^p = \langle S^p, I^p, F^p, A^p, \Delta^p \rangle$

Recall:

The transition relation $(s, x, w) \xrightarrow{a} (s', x', w')$ is defined by

- exactly one process moves
- counter actions behave as expected
- communication actions depend on the endpoint’s type

$\bullet \xrightarrow{c} c! \equiv c!x$

$\circ \xrightarrow{c} c! \equiv c!x ; x \equiv \text{any}$

$\bullet \xrightarrow{c} c? \equiv c?x$

$\circ \xrightarrow{c} c? \equiv x \equiv \text{any} ; c?x$

$\circ$ Note: can be simulated by $\bullet$
**Parametrized Reachability Problem**

**Definition:**
Given a topology $\mathcal{T}$, the decision problem $\text{Rp-Sc1cm}(\mathcal{T})$ is

**Input:** a system of communicating one-counter machines $\mathcal{S}$ with topology $\mathcal{T}$

**Output:** whether there exists a full run in $[\mathcal{S}]$

A run $(s, x, w) \xrightarrow{*} (s', x', w')$ is full when $\begin{cases} (s, x, w) \text{ is initial} \\ (s', x', w') \text{ is final} \end{cases}$
Parametrized Reachability Problem

Definition:
Given a topology $\mathcal{T}$, the decision problem $\text{RP-Sc1CM}(\mathcal{T})$ is

**Input:** a system of communicating one-counter machines $\mathcal{S}$ with topology $\mathcal{T}$

**Output:** whether there exists a full run in $[\mathcal{S}]$

A run $(s, x, w) \xrightarrow{*} (s', x', w')$ is full when

\[
\begin{cases} 
(s, x, w) \text{ is initial} \\
(s', x', w') \text{ is final}
\end{cases}
\]

Goal:
Characterize the topologies $\mathcal{T}$ where $\text{RP-Sc1CM}(\mathcal{T})$ is decidable.
Main Result

Simple Undirected Cycle

Simple Undirected Shunt

Theorem: \( R_p - S_{c1 \text{cm}} (T) \) is decidable if \( T \) is cycle-free and shunt-free.
Main Result

Simple Undirected Cycle

Simple Undirected Shunt

Theorem: \( \text{RP-Sc1CM}(\mathcal{T}) \) is decidable iff \( \mathcal{T} \) is cycle-free and shunt-free

\( \Rightarrow \) cycle-free: no simple undirected cycle
\( \Rightarrow \) shunt-free: no simple undirected shunt
Motivating Example: Sliding Window Protocol

Systems of Communicating One-Counter Machines and their Topology Parametrized Reachability Problem

Main Theorem: Proof of “Only If” Direction

Main Theorem: Proof of “If” Direction

Related/On-going/Future Work
Reduce known Undecidability Results

Simulation of communicating finite-state machines (CFSM)

- encode finite message exchange in exchange of counters
- reduce undecidability of reachability on cyclic architectures
Reduce known Undecidability Results

Idea:
Simulation of communicating finite-state machines (CFSM)

- encode finite message exchange in exchange of counters
- reduce undecidability of reachability on \textit{cyclic} architectures

Idea:
Simulation of two-counters Minsky machines

- reduce undecidability of reachability on simple \textit{shunt}
Agenda

- Motivating Example: Sliding Window Protocol
- Systems of Communicating One-Counter Machines and their Topology Parametrized Reachability Problem
- Main Theorem: Proof of “Only If” Direction
- Main Theorem: Proof of “If” Direction
- Related/On-going/Future Work
Cycle-free & Shunt-free Topologies

Form of topologies that are weakly-connected, cycle-free and shunt-free

- Two “roots” $r_1 \rightarrow r_2$
- Every simple undirected path from $\{r_1, r_2\}$ to $p \notin \{r_1, r_2\}$ ends with $\cdots \leftarrow p$
Cycle-free & Shunt-free Topologies

Form of topologies that are weakly-connected, cycle-free and shunt-free

- Two “roots” \( r_1 \rightarrow r_2 \)
- Every simple undirected path from \( \{r_1, r_2\} \) to \( p \not\in \{r_1, r_2\} \) ends with \( \cdots \rightarrow o \ p \)

[ Recall: \( o \) can be simulated by \( \bullet \) ]
Case of Two Processes

\[ p \xrightarrow{c} q \]

Reachability is undecidable for the class of one-counter machines with Presburger-definable updates.
Case of Two Processes

Intersect reachability relations of $p$ and $q$ between synchronizations.

Idea:

$$c!$$

$$c?$$

$$(s, x) \overset{\tau}{\rightarrow} (t, y) | \{z\}$$

$$(s, t)$$

Presburger's $2$nd

Reachability is undecidable for the class of one-counter machines with Presburger-definable updates.
Case of Two Processes

Idea:
Intersect reachability relations of $p$ and $q$ between synchronizations

\[ \ldots \xrightarrow{c!c?} (s^p, s^q) \xrightarrow{\chi_{s,t}} (t, y) \xrightarrow{*} (t^p, t^q) \xrightarrow{c!c?} \ldots \]

Reachability is undecidable for the class of one-counter machines with Presburger-definable updates
Case of Two Processes

**Idea:**

Intersect reachability relations of $p$ and $q$ between synchronizations

\[ \chi_{s,t}(x, y) = (s^p, x) \xrightarrow{*} (t^p, y) \land (s^q, x) \xrightarrow{*} (t^q, y) \in \text{Presb}_2 \]
Case of Two Processes

**Idea:**
Intersect reachability relations of \( p \) and \( q \) between synchronizations

\[
\chi_{s,t}(x, y) = (s^p, x) \xrightarrow{p} (t^p, y) \land (s^q, x) \xrightarrow{q} (t^q, y) \in \text{Presb}_2
\]

**Issue:**
Reachability is **undecidable** for the class of one-counter machines with Presburger-definable updates
Fix two distinguished Presburger variables $x$ and $y$

The class of one-counter Presburger predicates is generated by

$$\psi ::= \varphi(x) \mid \varphi(y) \mid \varphi(x - y) \mid \varphi(y - x) \mid \psi \wedge \psi \mid \psi \vee \psi \mid \top \mid \bot$$

where $\varphi$ ranges over unary Presburger predicates

Theorem: $s, t (x, y)$ can be translated into a one-counter machine $X$
Fix two distinguished Presburger variables $x$ and $y$

The class of one-counter Presburger predicates is generated by

$$\varphi \::= \varphi(x) \mid \varphi(y) \mid \varphi(x - y) \mid \varphi(y - x) \mid \varphi \land \psi \mid \varphi \lor \psi \mid \top \mid \bot$$

where $\varphi$ ranges over unary Presburger predicates

**Theorem:**

For every binary relation $R \subseteq \mathbb{N} \times \mathbb{N}$, the two following assertions are equivalent:

1) $R = \{(x, y) \mid (s, x) \xrightarrow{*}(t, y)\}$ for some one-counter machine

2) $R = \llbracket \psi \rrbracket$ for some one-counter Presburger predicate $\psi$

$\Rightarrow \chi_{s,t}(x, y)$ can be translated into a one-counter machine $\checkmark$
Idea:
- Schedule \( q \) last: \( q \) moves only when \( p \) attempts to receive from \( c \).
- Communications between \( p \) and \( q \) become synchronizations.
- States of \( p \) become pairs \((s_p, s_q)\).
- Rules \((s_p, c?, t_p)\) of \( p \) become \(((s_p, s_q), \text{test}(\cdot))\), \(((t_p, t_q))\) where \( \text{test}(\cdot) = \theta \cdot (u, c!', t_q) \cdot q^s q^z \cdot ! q(u, x) \).

Use Presburger-definability of post-\( \! \) for one-counter machines.
Merging Leaf Processes

Idea:

Merge leaf process $q$ into $p$ by summarizing $q$’s behavior.

Schedule $q$ last: $q$ moves only when $p$ attempts to receive from $c$.

Communications between $p$ and $q$ become synchronizations $c \cdot c$.

States of $p$ become pairs $(s_p, s_q)$.

Rules of $p$ become $(s_p, c? t_p, t_q)$ where $t = u \cdot (u, c!, t_q) \cdot q(s_q, z) \cdot ! q(u, x)$.

Use Presburger-definability of post $\overset{\cdot}{\cdot}$ for one-counter machines.
Merging Leaf Processes

Idea:
Merge leaf process $q$ into $p$ by summarizing $q$’s behavior

- Schedule $q$ last: $q$ moves only when $p$ attempts to receive from $c$
Merging Leaf Processes

Idea:

Merge leaf process q into p by summarizing q’s behavior

- Schedule q last: q moves only when p attempts to receive from c
- Communications between p and q become synchronizations c! ⋅ c?
Merging Leaf Processes

Idea:
Merge leaf process $q$ into $p$ by summarizing $q$’s behavior

- Schedule $q$ last: $q$ moves only when $p$ attempts to receive from $c$
- Communications between $p$ and $q$ become synchronizations $c! \cdot c?$
- States of $p$ become pairs $(s^p, s^q)$
Merging Leaf Processes

Idea:
Merge leaf process \( q \) into \( p \) by summarizing \( q \)'s behavior

- Schedule \( q \) last: \( q \) moves only when \( p \) attempts to receive from \( c \)
- Communications between \( p \) and \( q \) become synchronizations \( c ! \cdot c ? \)
- States of \( p \) become pairs \((s^p, s^q)\)
- Rules \((s^p, c ? , t^p)\) of \( p \) become \((((s^p, s^q), \text{test}(\varphi)), (t^p, t^q))\) where

\[
\varphi = \exists u \exists z \cdot (u, c !, t^q) \in \Delta^q \land (s^q, z) \xrightarrow{*} q (u, x)
\]
Merging Leaf Processes

Idea:

Merge leaf process $q$ into $p$ by summarizing $q$'s behavior

- Schedule $q$ last: $q$ moves only when $p$ attempts to receive from $c$
- Communications between $p$ and $q$ become synchronizations $c! \cdot c?$
- States of $p$ become pairs $(s^p, s^q)$
- Rules $(s^p, c?, t^p)$ of $p$ become $((s^p, s^q), \text{test}(\varphi), (t^p, t^q))$ where

$$
\varphi = \exists u \exists z \cdot (u, c!, t^q) \in \Delta^q \land (s^q, z) \xrightarrow{\varphi} (u, x)
$$

- Use Presburger-definability of $post^*$ for one-counter machines
Agenda

- Motivating Example: Sliding Window Protocol
- Systems of Communicating One-Counter Machines and their Topology Parametrized Reachability Problem
- Main Theorem: Proof of “Only If” Direction
- Main Theorem: Proof of “If” Direction
- Related/On-going/Future Work
Summary

- Formal model of Systems of Communicating One-Counter Machines
Summary

- Formal model of Systems of Communicating One-Counter Machines
- Their topology-parametrized reachability question
Summary

- Formal model of Systems of Communicating One-Counter Machines
- Their topology-parametrized reachability question
- Complete characterization of decidability in terms of topologies
Summary

- Formal model of Systems of Communicating One-Counter Machines
- Their topology-parametrized reachability question
- Complete characterization of decidability in terms of topologies
- Technical (side-)result: reachability relations of one-counter machines fall in “good” fragment of Presburger arithmetics
Related Works

- communicating finite state machines (CFSM) [Brand/Zafiropoulo ’81, Pachl ’82]
- characterizing decidable topologies for mixed lossy & reliable channels [Chambart/Schnoebelen ’08]
- CFSM with infinite message alphabets [Le Gall/Jeannet ’07]
- (fifo-) communicating pushdown machines [LaTorre/Parlato/Madhusudan ’08]
- the influence of topologies on decidability [Heußner/Leroux/Muscholl/Sutre ’10]
- decidability restrictions of multi-counter machines, model checking register machines
Related Works

- communicating finite state machines (CFSM) [Brand/Zafiropoulo ’81, Pachl ’82]

- characterize decidable topologies for mixed lossy & reliable channels [Chambart/Schnoebelen ’08]
Related Works

- communicating finite state machines (CFSM) [Brand/Zafiropoulo ’81, Pachl ’82]
- characterize decidable topologies for mixed lossy & reliable channels [Chambart/Schnoebelen ’08]
- CFSM with infinite message alphabets [Le Gall/Jeannet ’07]
Related Works

- communicating finite state machines (CFSM) [Brand/Zafiropoulo ’81, Pachl ’82]
- characterize decidable topologies for mixed lossy & reliable channels [Chambart/Schnoebelen ’08]
- CFSM with infinite message alphabets [Le Gall/Jeannet ’07]
- (fifo-) communicating pushdown machines [LaTorre/Parlato/Madhusudan ’08]

and the influence of topologies on decidability [Heußner/Leroux/Muscholl/Sutre ’10]
Related Works

- communicating finite state machines (CFSM) [Brand/Zafiropoulo ’81, Pachl ’82]
- characterize decidable topologies for mixed lossy & reliable channels [Chambart/Schnoebelen ’08]
- CFSM with infinite message alphabets [Le Gall/Jeannet ’07]
- (fifo-) communicating pushdown machines [LaTorre/Parlato/Madhusudan ’08]

and the influence of topologies on decidability [Heußner/Leroux/Muscholl/Sutre ’10]

- decidable restrictions of multi-counter machines, model checking register machines, . . .
Decidability of Eager Reachability

Definition:
A full run $\rho$ is eager if matching $(c!, c?)$ pairs are consecutive in $\rho$

If $T$ is cycle-free,
- Full runs can be re-ordered into eager ones
- $\text{RP-Sc1CM-Eager}(T)$ is decidable iff $T$ is shunt-free

Proposition:
If $T$ is strongly connected, then $\text{RP-Sc1CM-Eager}(T)$ is decidable iff $T$ contains at most two processes

Open: full characterization of decidable topologies (for eager reachability)
Perspectives

**Complexity** of $\mathbb{RP}$-$SC_1CM$ for decidable topologies

- At least PSPACE-hard
Perspectives

**Complexity** of $\text{RP-Sc}1_{\text{CM}}$ for decidable topologies

- At least PSPACE-hard

**Lossy channel** Communicating One-Counter Machines

- Undecidable for using acknowledgments
Perspectives

**Complexity** of RP-Sc1cm for decidable topologies

- At least PSPACE-hard

**Lossy channel** Communicating One-Counter Machines

- Undecidable for \[ \square \bullet \leftrightarrow \square \circ \circ \leftrightarrow \bullet \square \] using acknowledgments

Extension from counters to stacks (i.e., send/receive stack)

**Conjecture:**

\[ \text{RP-ScPDM}(T) \text{ is decidable} \quad \text{iff} \quad T \text{ is cycle-free and shunt-free} \]