MPass: An Efficient Tool for the Analysis of Message-Passing Programs

P. A. Abdulla[1] M. F. Atig[1] J. Cederberg[1] S. Modi[3]
O. Rezine[1] G. Saini[2]

Uppsala University, Sweden

Indian Institute of Technology, Ropar

Indian Institute of Technology, Kanpur

FACS 2014: The 11th International Symposium on Formal Aspects of Component Software

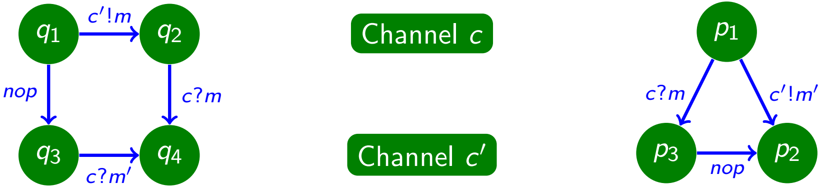
September 11, 2014

Summary

Summary

Background & Motivation: Message Passing Programs

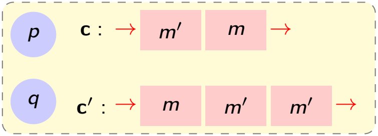
- Finite number of processes communicating via message passing.
  - Finite set of messages
- A process is modelled as finite-state (or pushdown) systems
  - Send operations
  - Receive operations
- Processes communicate over unbounded channels



Background & Motivation: Configuration

Configurations consist of:

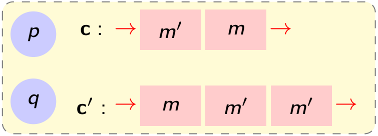
- Processes states: $q \in Q = \{(q_i, p_i)\}$
- Channel contents $\mathbf{w} \in (\Sigma^*)^C$, $\Sigma = \{m, m'\}$



Background & Motivation: Reachability

Configurations consist of:

- Processes states: $q \in Q = \{(q_i, p_i)\}$
- Channel contents $\mathbf{w} \in (\Sigma^*)^C$, $\Sigma = \{m, m'\}$

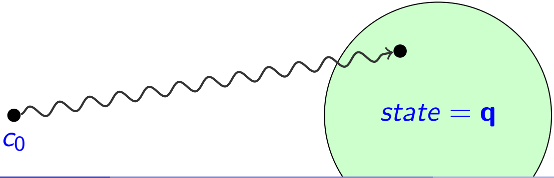


Definition (The State Reachability Problem)

Given:

- Message Passing Program
- Initial configuration conf_0
- Target state $\mathbf{q} = (q_{\text{target}}, p_{\text{target}})$

Is there a run from conf_0 to some configuration in which $\mathit{state} = \mathbf{q}$



Background & Motivation: Channel Semantics

- Perfect FIFO channels
  - Turing powerful model
    (even for one finite-state process and one channel)
- Lossy FIFO channels
  - Reachability is decidable but non-primitive recursive for finite-state processes [Abdulla et al. 93], [Schnoebelen 02]
  - Turing powerful model for pushdown processes
- Unordered channels (Multisets)
  - Reachability is decidable but EXPSPACE-hard for finite-state processes [Lipton 76], [Rackoff 78], [Mayr 81]
  - Turing powerful model for pushdown processes

Background & Motivation: Questions of Interests

- Find a decidable subclass for asynchronous communication protocols
- Use this subclass for approximate analysis:
  - Over-approximation: General model $\xrightarrow{\textit{Abstraction}}$ Decidable model s.t.:

    $\textit{Behaviors}(\textit{General model}) \subseteq \textit{Behaviours}(\textit{Decidable model})$

    Prove correctness

  - Under-approximation: General model $\xrightarrow{\textit{Restriction}}$ Decidable model s.t.:

    $\textit{Behaviours}(\textit{Decidable model}) \subseteq \textit{Behaviours}(\textit{General model})$

    Find errors

Background & Motivation: Questions of Interests

- Find a decidable subclass for asynchronous communication protocols
- Use this subclass for approximate analysis:
  - Under-approximation: General model $\xrightarrow{\text{Restriction}}$ Decidable model s.t.:

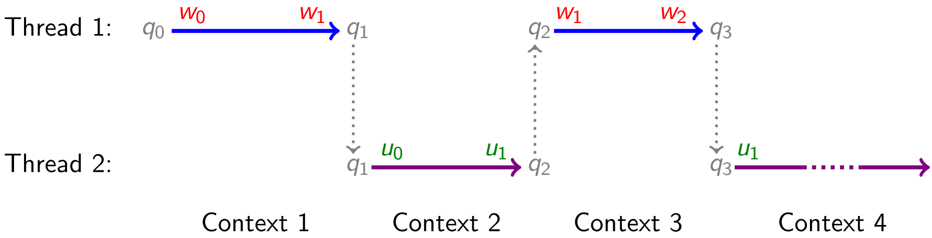
$$\textit{Behaviours}(\textit{Decidable model}) \subseteq \textit{Behaviours}(\textit{General model})$$

Find errors

Background & Motivation: Bounded Context-Switches as a Starting Point

Program class: Concurrent programs with a fixed number of processes communicating via shared variables

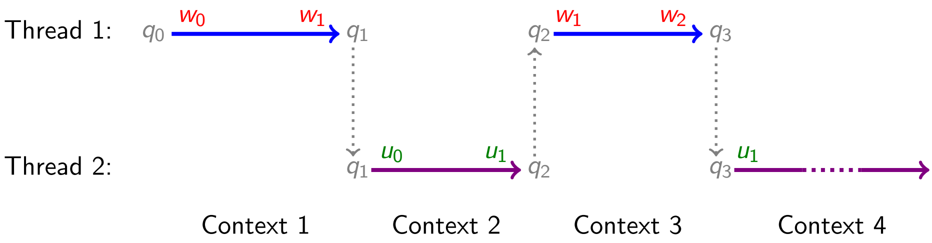
The idea ([Qadeer et al., 05]]): Analysis techniques based on bounding the number of context switches (interleaving) between threads



Background & Motivation: Bounded Context-Switches as a Starting Point

Program class: Concurrent programs with a fixed number of processes communicating via shared variables

The idea ([Qadeer et al., 05]]): Analysis techniques based on bounding the number of context switches (interleaving) between threads



$\Longrightarrow$ The bounded-context reachability problem is decidable.

Why Context-Bounded Analysis?

- Context-bounded algorithm is suitable for the analysis of concurrent programs communicating via shared variables.
- Experiments: Many subtle concurrency errors are manifested in executions with a small number of contexts.
- Several implementations: CHESS, jMoped, ...
- Complexity of safety properties verification:

	Unbounded	Context-bounded
Finite-state systems	PSPACE-complete	NP-complete
Pushdown systems	Undecidable	NP-complete

Why (Not) Context-Bounded Analysis?

Problem, in the case of message passing programs:

- Bounding the number of context switches will not affect the decidability/complexity of the verification problems in general

Consequence:

- This approach is inadequate for communication protocols

Solution:

- Adapt the notion of context to message passing programs.

Summary

Proposed Approach: From Contexts to Phases

From **contexts** to **phases**:

- A phase describes the communication behaviour of a process.
- A process goes through successive communication phases during a system run.

Proposed **definitions**:

[La Torre et al. 2008]

Both:

(1) { Receive messages from only one channel }, and

(2) { Send messages to any other channel }

[Abdulla et al. 2013]

Either:

(1) { Send messages from any channel }, or

(2) { Receive messages from any channel }

but not both.

Proposed Approach: From Contexts to Phases

From **contexts** to **phases**:

- A phase describes the communication behaviour of a process.
- A process goes through successive communication phases during a system run.

Proposed **definitions**:

[La Torre et al. 2008]

Both:

(1) { Receive messages from only one channel }, and

(2) { Send messages to any other channel }

[Abdulla et al. 2013]

Either:

(1) { Send messages from any channel }, or

(2) { Receive messages from any channel }

but not both.

Proposed Approach: From Contexts to Phases

From **contexts** to **phases**:

- A phase describes the communication behaviour of a process.
- A process goes through successive communication phases during a system run.

Proposed **definitions**:

[La Torre et al. 2008]

Both:

(1) { Receive messages from only one channel }, and

(2) { Send messages to any other channel }

[Abdulla et al. 2013]

Either:

(1) { Send messages from any channel }, or

(2) { Receive messages from any channel }

but not both.

Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.

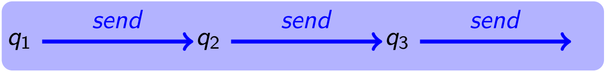
#Phases #Switches

Process 1:

Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.

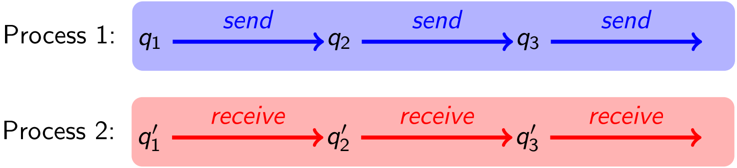
Process 1:



#Phases	#Switches
1	0

Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



#Phases	#Switches
1	0

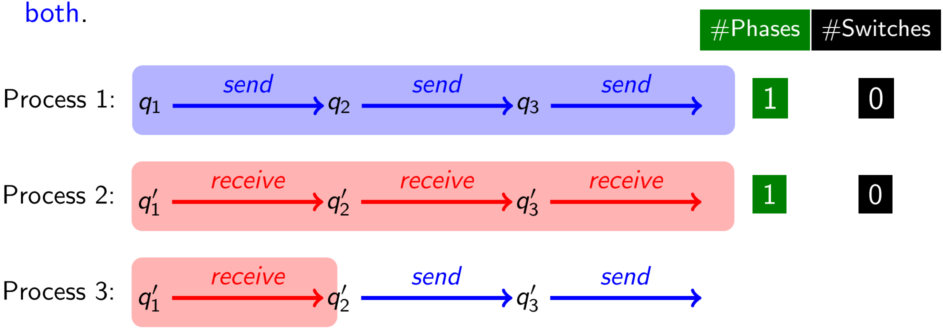
Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.

		#Phases	#Switches
Process 1:	$q_1 \xrightarrow{send} q_2 \xrightarrow{send} q_3 \xrightarrow{send}$	1	0
Process 2:	$q'_1 \xrightarrow{receive} q'_2 \xrightarrow{receive} q'_3 \xrightarrow{receive}$	1	0

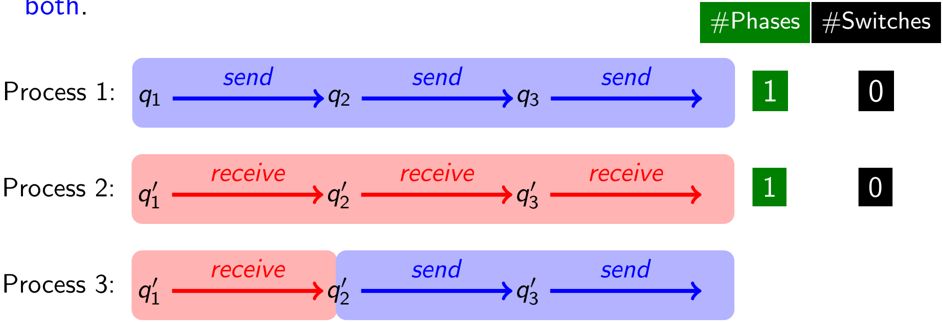
Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



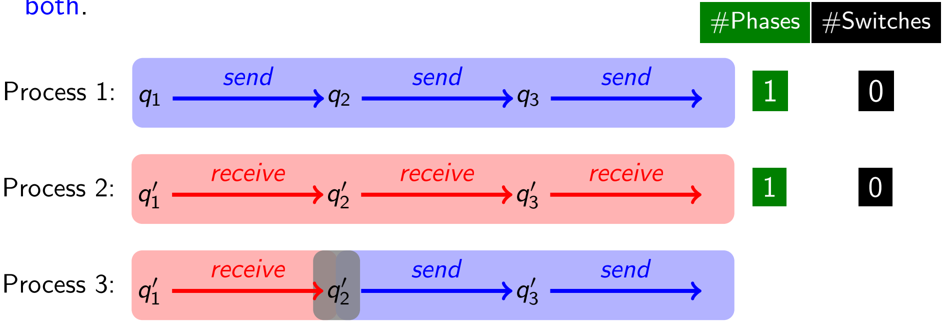
Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



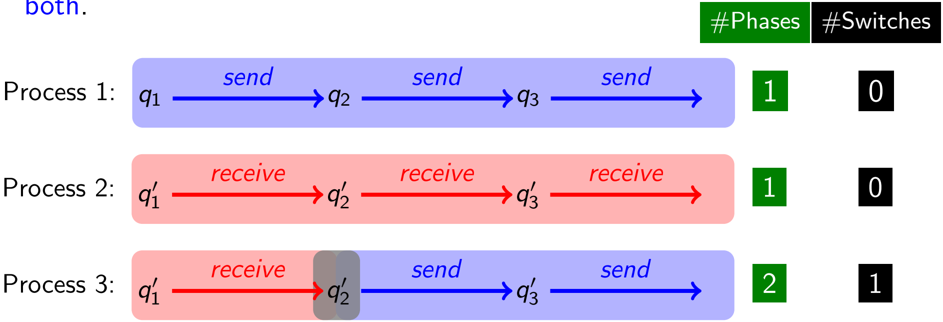
Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



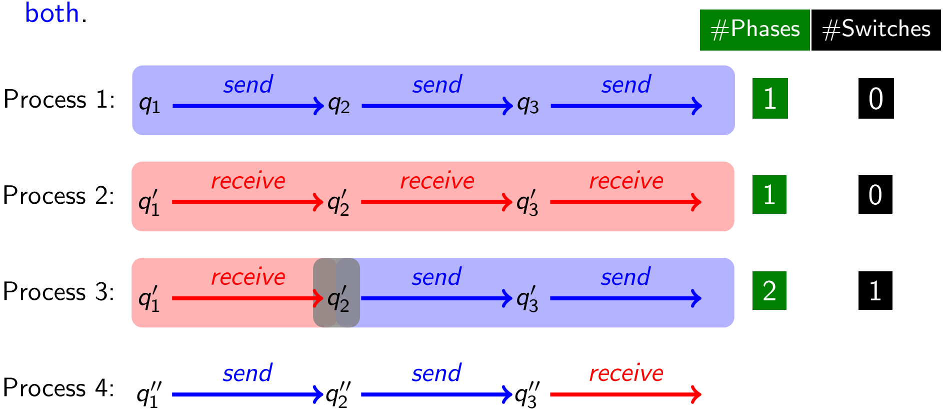
Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



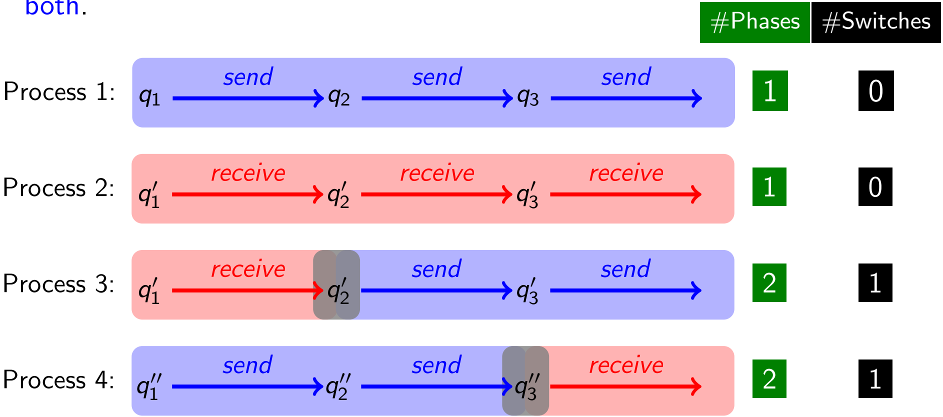
Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



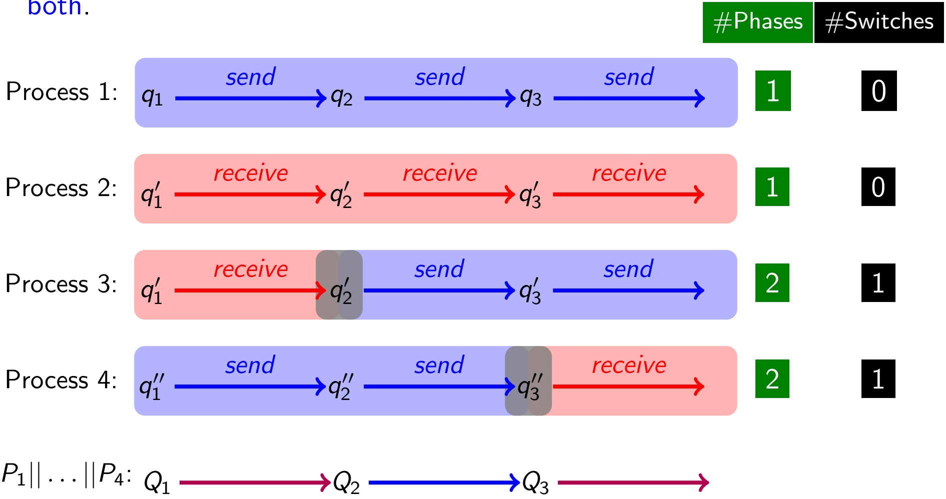
Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



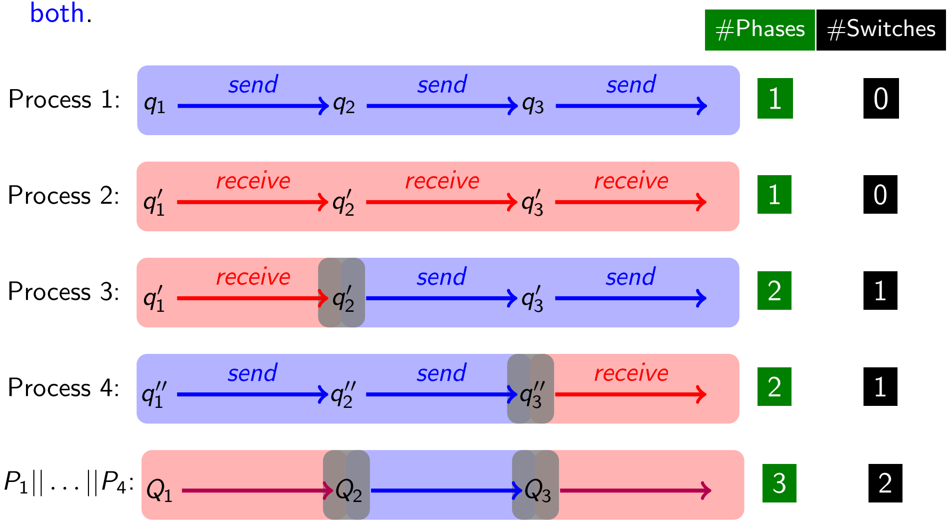
Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



Proposed Approach: Bounded Phase Analysis

In a **phase**, a process performs either send or receive operations, but not both.



Proposed Approach: Bounded Phase Analysis

Advantages:

- Unbounded computation within each phase.
- Unbounded channel capacity
- Unbounded number of switches between processes

Proposed Approach: Bounded Phase Analysis

Advantages:

- Unbounded computation within each phase.
- Unbounded channel capacity
- Unbounded number of switches between processes
- Better complexity . . .

Proposed Approach: Complexity

- Finite-State Processes:

	Unbounded	Phase-bounded
Perfect FIFO Channels	Undecidable	Undecidable
Lossy FIFO Channels	Non-primitive Recursive	NP-complete
Unordered Channels	EXPSPACE-hard	NP-complete

- PushDown Processes:

	Unbounded	Phase-bounded
Perfect FIFO Channels	Undecidable	Undecidable
Lossy FIFO Channels	Undecidable	Undecidable
Unordered Channels	Undecidable	NP-complete

Proposed Approach: Complexity

- Finite-State Processes:

	Unbounded	Phase-bounded
Perfect FIFO Channels	Undecidable	Undecidable
Lossy FIFO Channels	Non-primitive Recursive	NP-complete
Unordered Channels	EXPSPACE-hard	NP-complete

- PushDown Processes:

	Unbounded	Phase-bounded
Perfect FIFO Channels	Undecidable	Undecidable
Lossy FIFO Channels	Undecidable	Undecidable
Unordered Channels	Undecidable	NP-complete

Summary

Lossy Channel Systems

Definition

A *Lossy Channel System* is a tuple $S = (n, Q, \Sigma, C, \Delta)$, where

n is the number of processes,

Q is the set of states,

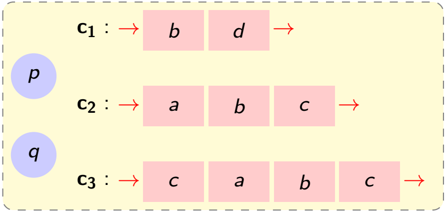
Σ is the message alphabet,

C is the set of channels, and

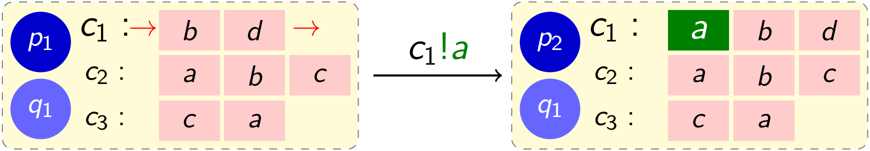
$\Delta \subseteq Q \times C \times \{!, ?\} \times \Sigma \times Q$ is the set of transition rules.

Configurations consist of:

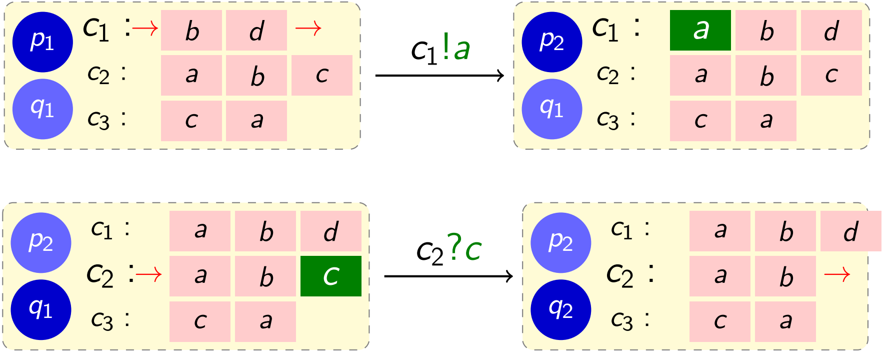
- A state $q_i \in Q$ for the process i
- Channel contents $\mathbf{w} \in (\Sigma^*)^C$



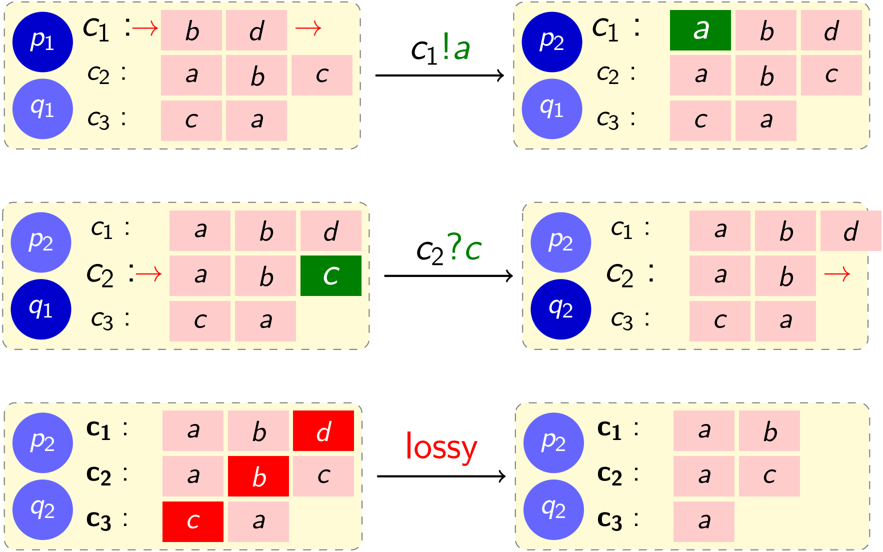
Lossy Channel Systems



Lossy Channel Systems



Lossy Channel Systems



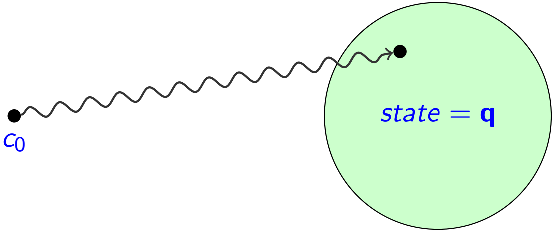
Reachability Problem

Definition (The State Reachability Problem)

Given:

- Lossy channel system S
- Initial configuration c_0
- Target states $\mathbf{q} = (q_1^{target}, q_2^{target}, \ldots, q_n^{target})$

Is there a run from c_0 to some configuration in which $state = \mathbf{q}$



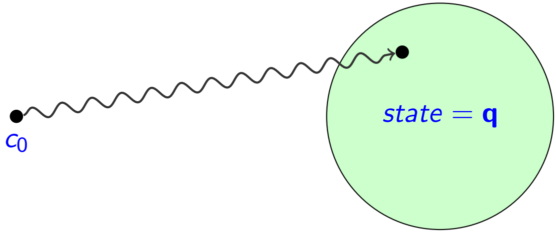
Bounded-Phase Reachability

Definition (The Bounded-Phase Reachability Problem)

Given:

- Lossy channel system S
- Phase bound k
- Initial configuration c_0
- Target states $\mathbf{q} = (q_1^{target}, q_2^{target}, \ldots, q_n^{target})$

Is there a run from c_0 to some configuration in which $state = \mathbf{q}$ and where each process performs at most k phases



Bounded-Phase Reachability

Definition (The Bounded-Phase Reachability Problem)

Given:

- Lossy channel system S
- Phase bound k
- Initial configuration c_0
- Target states $\mathbf{q} = (q_1^{target}, q_2^{target}, \ldots, q_n^{target})$

Is there a run from c_0 to some configuration in which $state = \mathbf{q}$ and where each process performs at most k phases

Theorem [Abdudlla et al. 2013]

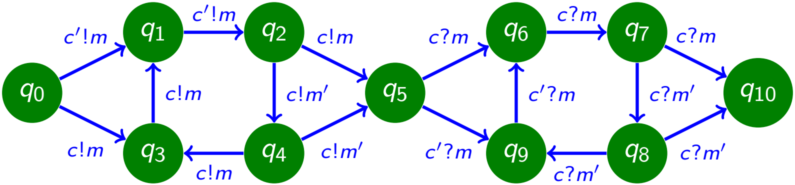
Bounded-phase reachability problem is polynomially reducible to the satisfiability of quantifier-free Presburger formulas.

Summary

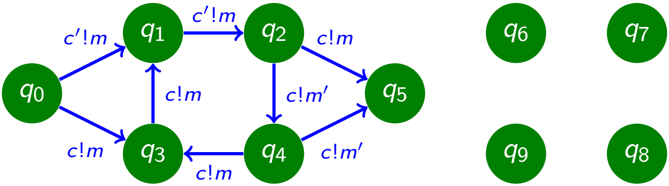
From Reachability to Satisfiability

Proof Idea: Send Copies (1/2)

- Let us assume that a process is defined by the following automaton

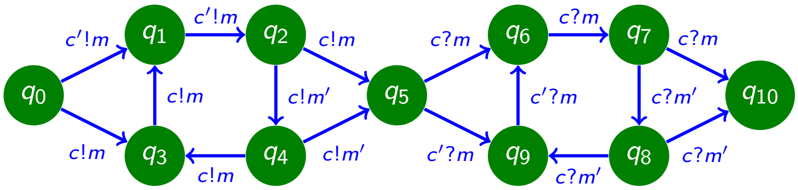


- The send copy is constructed by removing all receive transitions:

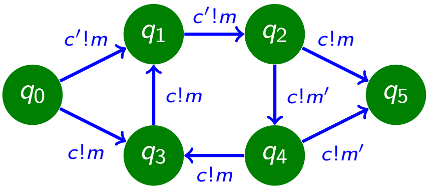


Proof Idea: Send Copies (1/2)

- Let us assume that a process is defined by the following automaton

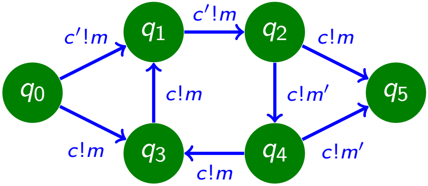


- The send copy is constructed by removing all receive transitions:

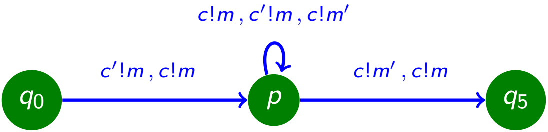


Proof Idea: Send Copies (2/2)

- The send copy is constructed by removing all receive transitions:

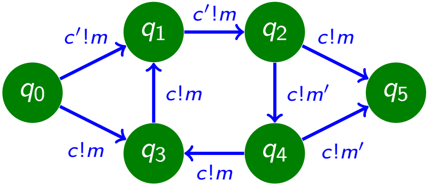


- Since messages can be lost, any SCC can be collapsed in one state

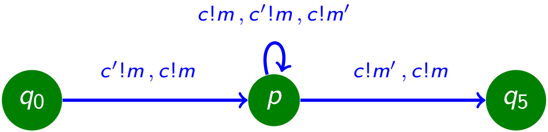


Proof Idea: Send Copies (2/2)

- The send copy is constructed by removing all receive transitions:



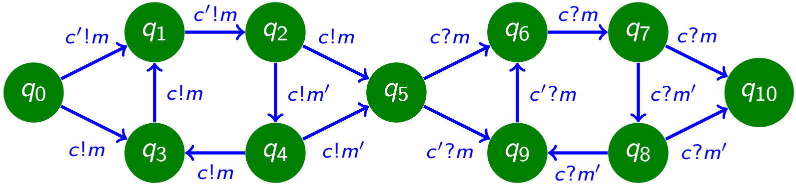
- Since messages can be lost, any SCC can be collapsed in one state



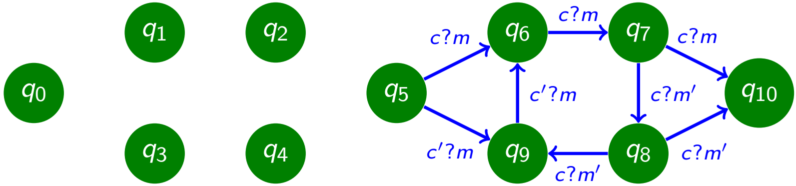
- The send copies contain only self-loops

Proof Ideas: Receive Copies

- Let us assume that a process is defined by the following automaton

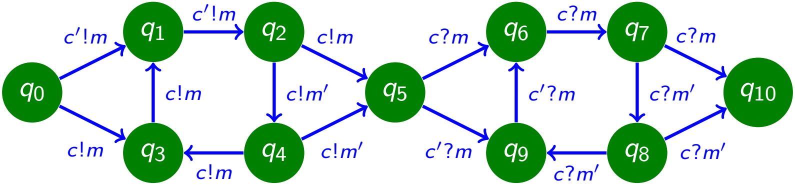


- The receive copy is constructed by removing all send transitions:

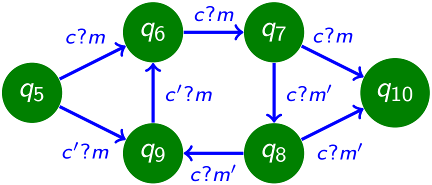


Proof Ideas: Receive Copies

- Let us assume that a process is defined by the following automaton

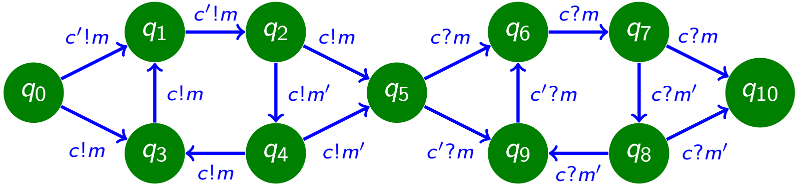


- The receive copy is constructed by removing all send transitions:

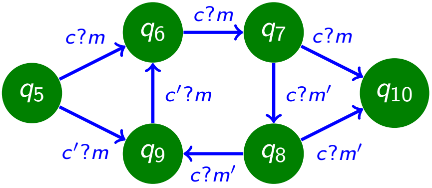


Proof Ideas: Receive Copies

- Let us assume that a process is defined by the following automaton

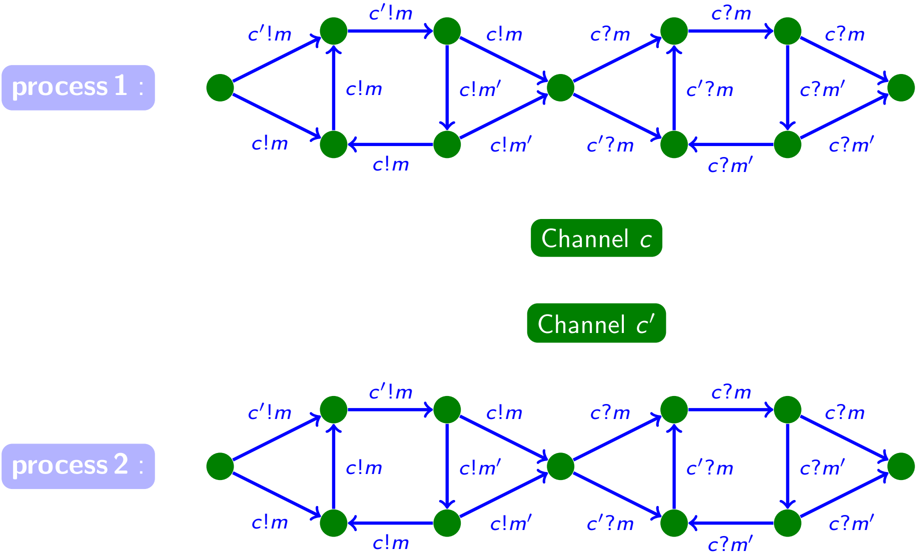


- The receive copy is constructed by removing all send transitions:



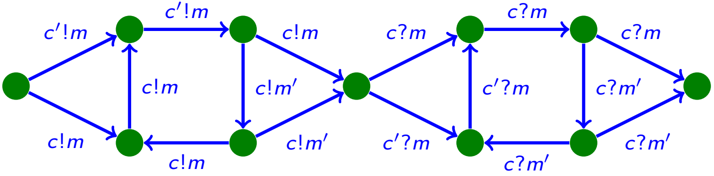
Since messages can be lost, we only need to consider only receive simple paths

Proof Ideas: Reduction to Simple Path Reachability



Proof Ideas: Reduction to Simple Path Reachability

process 1 :



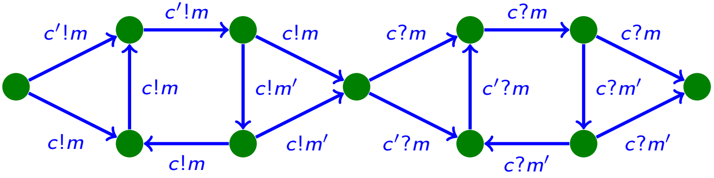
Given a bound k, for each process:

- Construct k send/receive copies
- Each send/receive copy represents a phase

Channel c

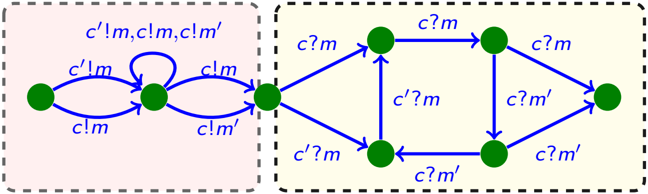
Channel c'

process 2 :



Proof Ideas: Reduction to Simple Path Reachability

process 1 :



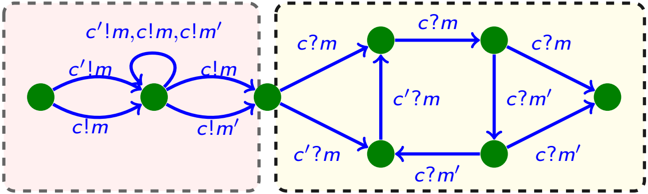
Given a bound k, for each process:

- Construct k send/receive copies
- Each send/receive copy represents a phase

Channel c

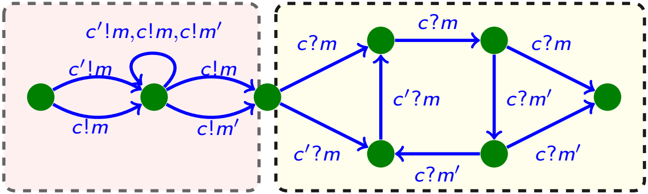
Channel c'

process 2 :



Proof Ideas: Reduction to Simple Path Reachability

process 1 :

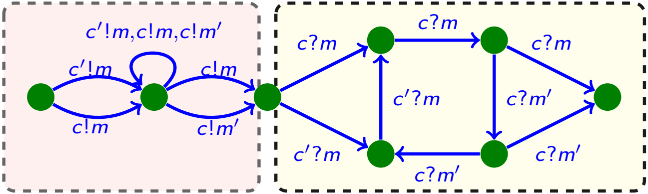


- The total number of received messages is bounded by $k|S|$ (since we consider only simple paths in the receive phases)
- Unfold the simple loops accordingly.

Channel c

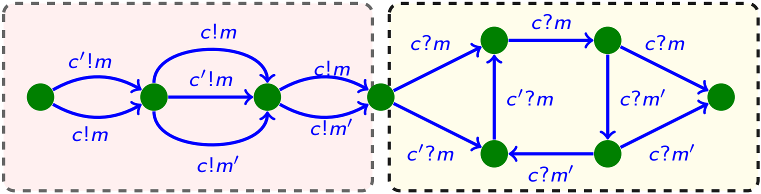
Channel c'

process 2 :



Proof Ideas: Reduction to Simple Path Reachability

process 1 :

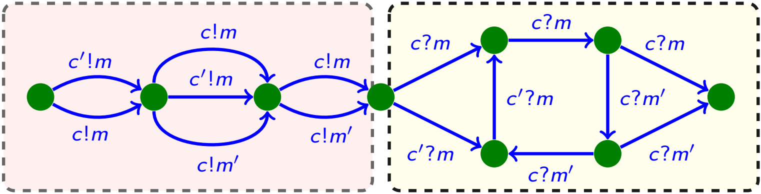


- The total number of received messages is bounded by $k|S|$ (since we consider only simple paths in the receive phases)
- Unfold the simple loops accordingly.

Channel c

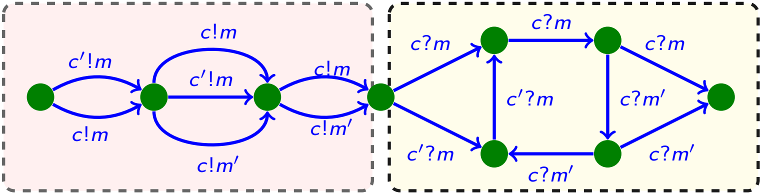
Channel c'

process 2 :



Proof Ideas: Reduction to Simple Path Reachability

process 1 :



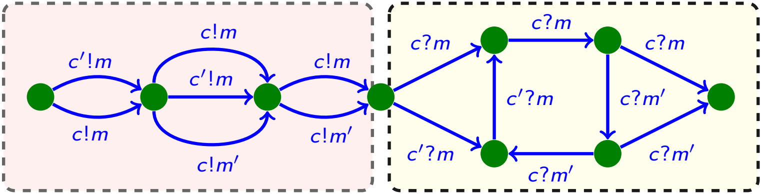
- The total number of received messages is bounded by $k|S|$ (since we consider only simple paths in the receive phases)
- Unfold the simple loops accordingly.

Channel c

Channel c'

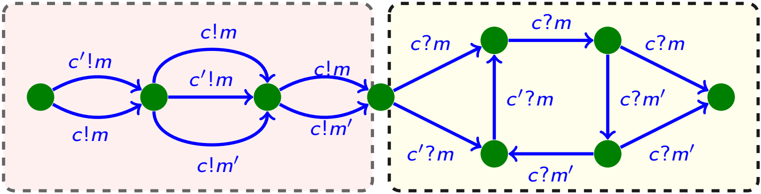
Only consider simple paths of the constructed processes

process 2 :



Proof Ideas: Quantifier-Free Formula Construction (1/2)

process 1 :

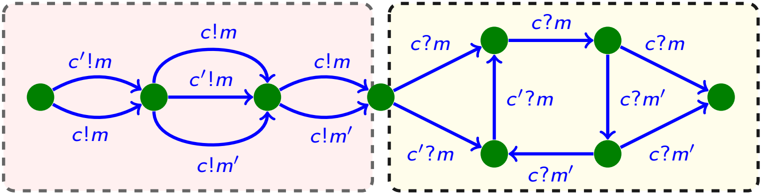


Channel c

- Write a formula to select a simple path in each process)

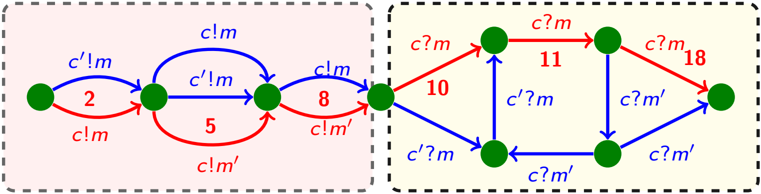
Channel c'

process 2 :



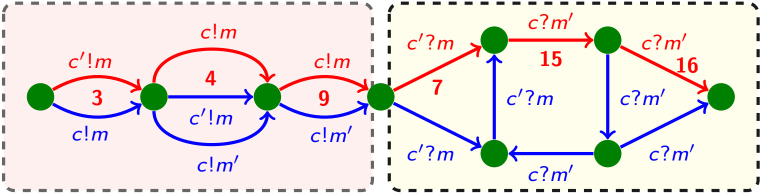
Proof Ideas: Quantifier-Free Formula Construction (1/2)

process 1 :



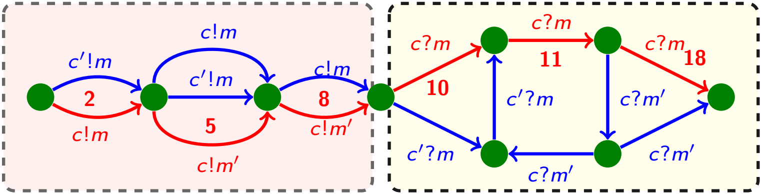
- For each transition t, we associate a unique variable $index(t)$
- A transition is executed iff $index(t) > 0$
- $index(t) \neq index(t')$ for all executed transitions$t \neq t'$
- For each state, there is exactly one excuted input transition and one output transition t' s.t. $index(t) < index(t')$

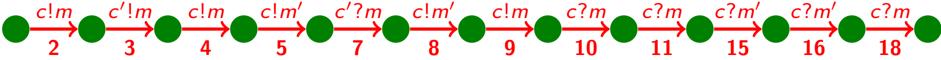
process 2 :



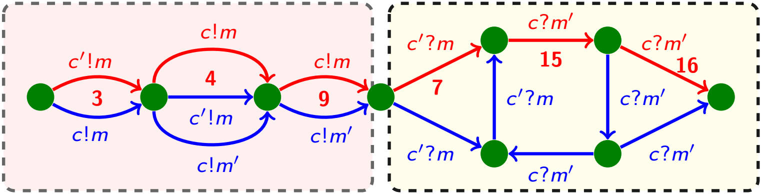
Proof Ideas: Quantifier-Free Formula Construction (1/2)

process 1 :

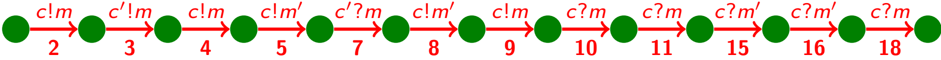




process 2 :

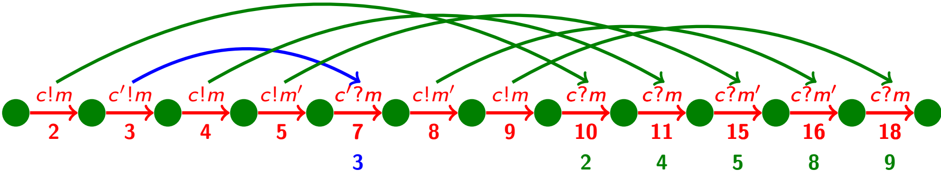


Proof Ideas: Quantifier-Free Formula Construction (2/2)



Write a formula to ensure that each receive transition is matched by a preceding send transition while respecting the channel semantics

Proof Ideas: Quantifier-Free Formula Construction (2/2)



- For each receive transition t, we associate a variable $match(t)$
- There is a matching send operation t' such that

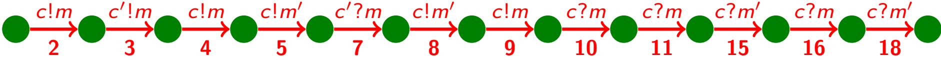
$$match(t) = index(t') < index(t)$$

- The lossy channel semantics is preserved for each channel:

$$0 < index(t_1) < index(t_2) \Rightarrow match(t_1) < match(t_2)$$

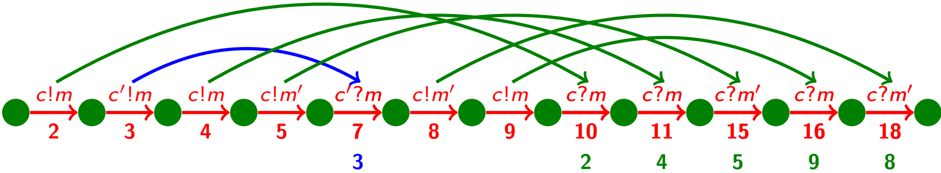
The Case of Unordered Channels

Quantifier-Free Formula Construction



Write a formula to ensure that each receive transition is matched by a preceding send transition while respecting the channel semantics

Quantifier-Free Formula Construction



- For each receive transition t, we associate a variable $match(t)$
- There is a matching send operation t' such that
  $$match(t) = index(t') < index(t)$$
- The lossy channel semantics is preserved for each channel:
  $$0 < index(t_1) \neq index(t_2) \Rightarrow match(t_1) \neq match(t_2)$$

Summary

Experimental Results

Experimental Results

P	Sem	Const. gen.	SMT	Total	Mod.	Res
ABP_F	SLCS	0.04	1	1.04	2	U
ABP_F	UCS	0.04	1	1.04	2	U
SlidingWindow_F	SLCS	0.02	0	0.02	1	U
SlidingWindow_F	UCS	0.03	0	0.03	1	U
Synchronous_F	SLCS	0.02	0	0.02	3	U
Synchronous_F	UCS	0.02	0	0.02	3	U
ABP	UCS	0.07	74	74.07	4	U
ABP	LCS	0.04	1	1.04	3	S
ABP	SLCS	0.03	2	2.03	3	S
STP	UCS	0.03	4	4.03	6	U
STP	LCS	0.02	0	0.02	4	S
STP	SLCS	0.02	0	0.02	4	S
Jingle	SLCS	18.4	10.8	19.2	8	U
Jingle	LCS	21.2	21.1	42.3	8	U

- An open source tool available at https://github.com/vigenere92/MPass
- Error are manifested with small number of phases

Conclusion and Future Work

Conclusion and Future Work

Conclusion

- A new concept for under-approximating message-passing protocols
- The framework can be instantiated to several classes of channel semantics
- An efficient reduction to the satisfiability problem for Quantifier-Free formulas
- An open source tool
- Errors are manifested with small number of phases

Semantics	Finite-state process	Pushdown process
Lossy	NP-COMPLETE	undecidable
Stuttering Lossy	NP-COMPLETE	undecidable
Unordered	NP-COMPLETE	NP-COMPLETE
Perfect	undecidable	undecidable

Table : Decidability/Complexity Results for the Bounded-Reachability

Conclusion and Future Work

Conclusion

- A new concept for under-approximating message-passing protocols
- The framework can be instantiated to several classes of channel semantics
- An efficient reduction to the satisfiability problem for Quantifier-Free formulas
- An open source tool
- Errors are manifested with small number of phases

Future Work

- Unbounded data-domain
- Over-approximation techniques

Thank you!!