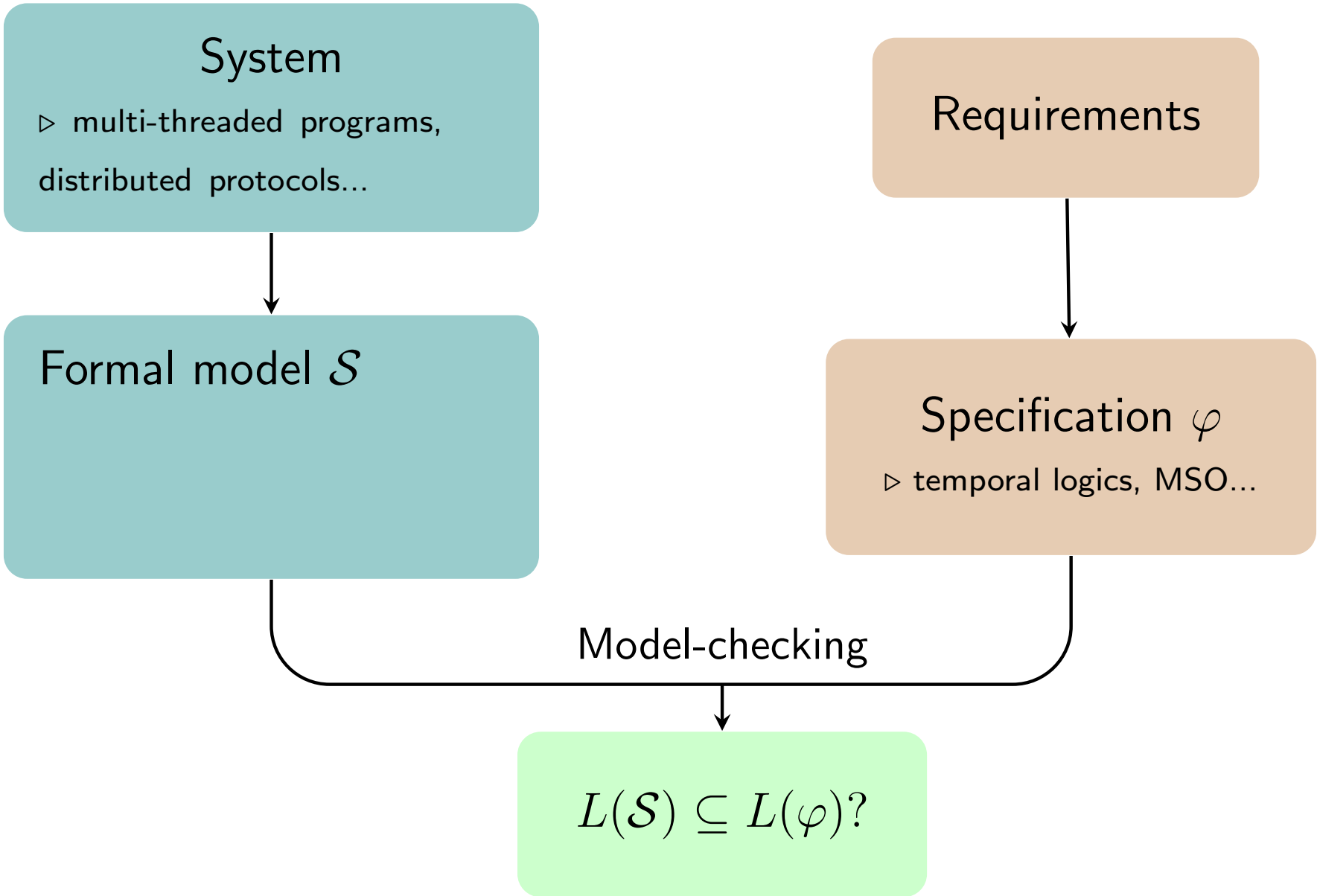
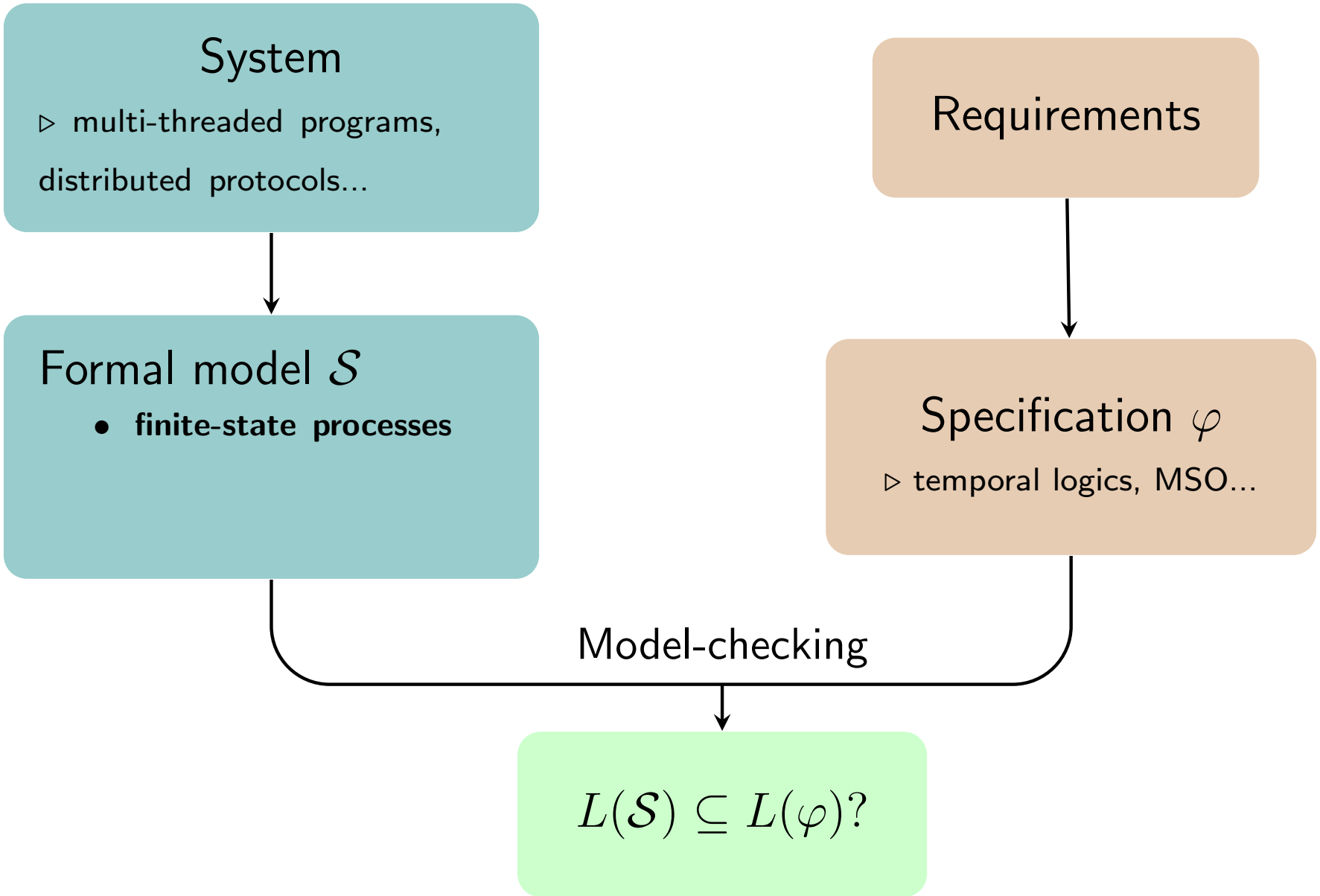
Verification of Parameterized Communicating Automata via Split-width

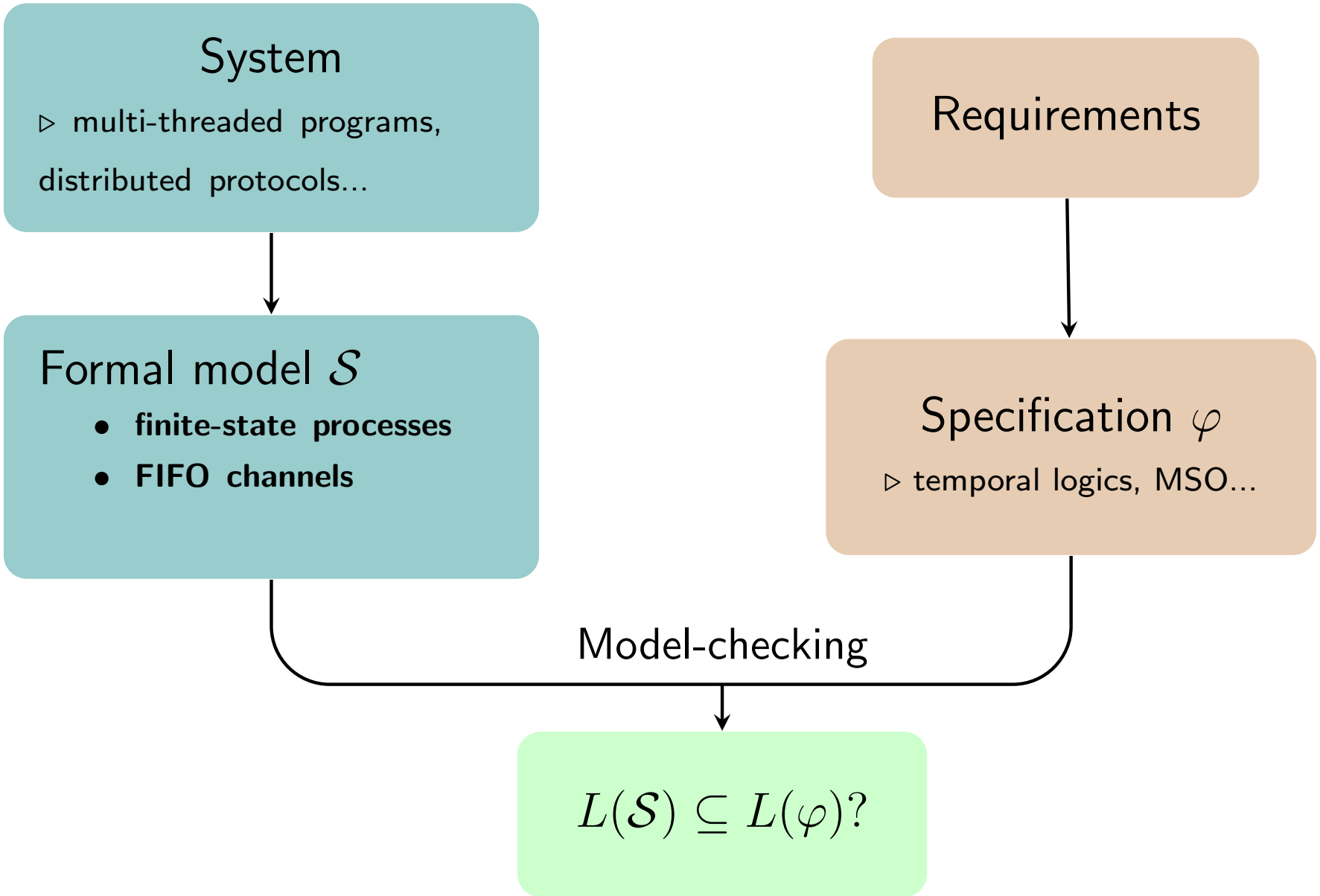
Marie Fortin, Paul Gastin

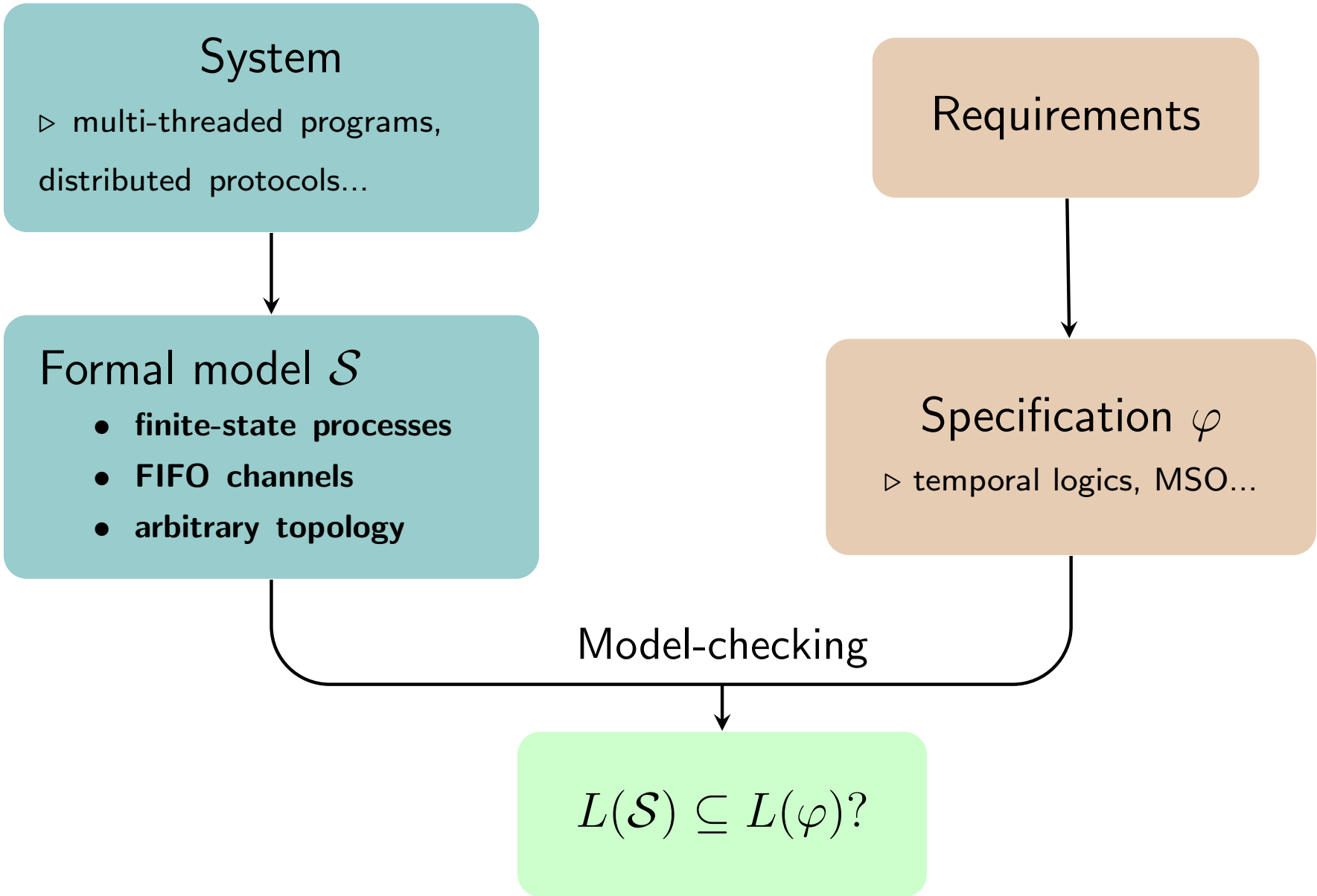
LSV, ENS Cachan

FoSSaCS 2016, Eindhoven


System
▷ multi-threaded programs,
distributed protocols...
Requirements
Formal model $\mathcal{S}$
Specification φ
▷ temporal logics, MSO...
Model-checking
$L(\mathcal{S}) \subseteq L(\varphi)$?


System
▷ multi-threaded programs,
distributed protocols...
Requirements
Formal model $\mathcal{S}$
• finite-state processes
Specification φ
▷ temporal logics, MSO...
Model-checking
$L(\mathcal{S}) \subseteq L(\varphi)$?

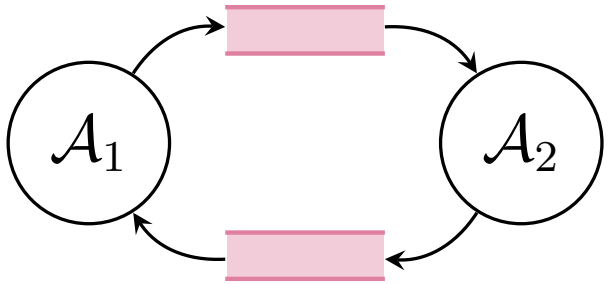

System
▷ multi-threaded programs,
distributed protocols...
Requirements
Formal model $\mathcal{S}$
• finite-state processes
• FIFO channels
Specification φ
▷ temporal logics, MSO...
Model-checking
$L(\mathcal{S}) \subseteq L(\varphi)$?


System
▷ multi-threaded programs,
distributed protocols...
Requirements
Formal model $\mathcal{S}$
• finite-state processes
• FIFO channels
• arbitrary topology
Specification φ
▷ temporal logics, MSO...
Model-checking
$L(\mathcal{S}) \subseteq L(\varphi)$?

Sources of undecidability

- Unbounded FIFO channels

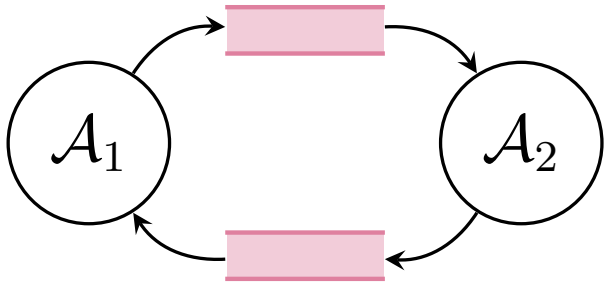
  $\rightarrow$ Undecidable even for 2 processes



Sources of undecidability

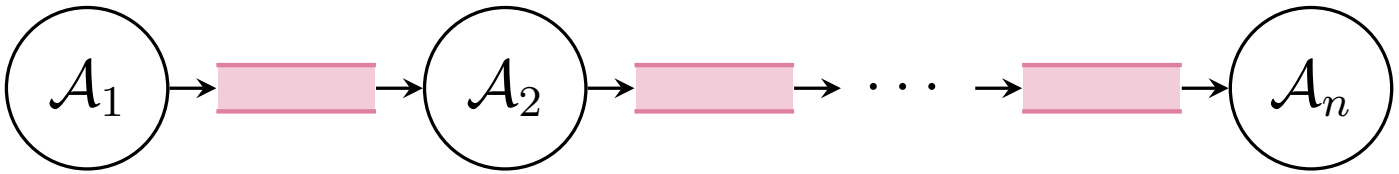
- Unbounded FIFO channels

  $\rightarrow$ Undecidable even for 2 processes



- Arbitrarily many processes

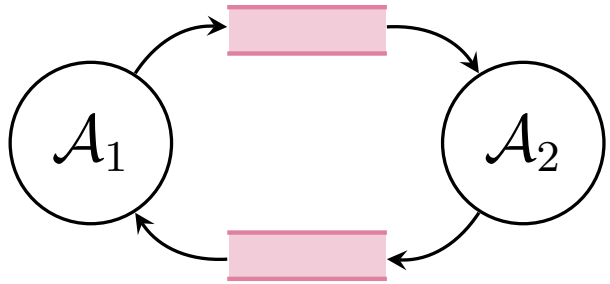
  $\rightarrow$ Undecidable even for bounded channels



Sources of undecidability

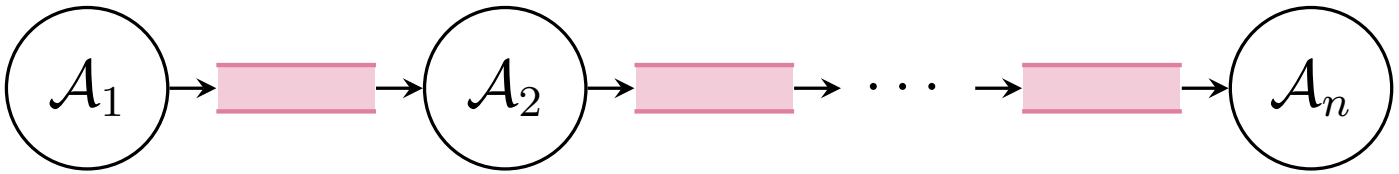
- Unbounded FIFO channels

  $\rightarrow$ Undecidable even for 2 processes



- Arbitrarily many processes

  $\rightarrow$ Undecidable even for bounded channels



- Arbitrary topology

  $\rightarrow$ Undecidable even for locally bounded runs

Goal

A generic approach to define

- decidable restrictions
- efficient decision procedures

using graph decompositions and reductions to tree-automata.

Goal

A generic approach to define

- decidable restrictions
- efficient decision procedures

using graph decompositions and reductions to tree-automata.

- P. Madhusudan, G. Parlato: The Tree Width of Auxiliary storage. POPL 2011.
- C. Aiswarya, P. Gastin, K. Narayan Kumar: Verifying Communicating Multi-pushdown Systems via Split-width. ATVA 2014.

Goal

A generic approach to define

- decidable restrictions
- efficient decision procedures

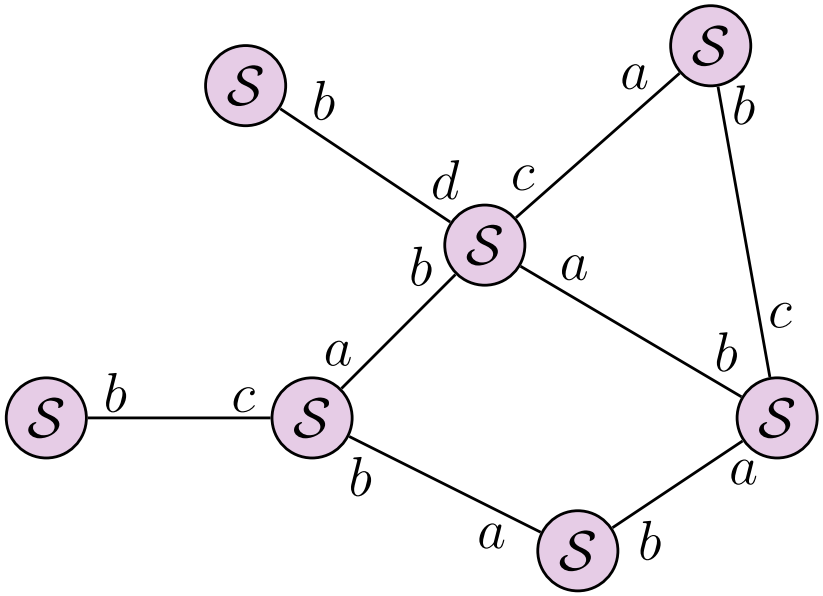
using graph decompositions and reductions to tree-automata.

- P. Madhusudan, G. Parlato: The Tree Width of Auxiliary storage. POPL 2011.
- C. Aiswarya, P. Gastin, K. Narayan Kumar: Verifying Communicating Multi-pushdown Systems via Split-width. ATVA 2014.

New: arbitrary topology, unknown number of processes

Parameterized Communicating Automata

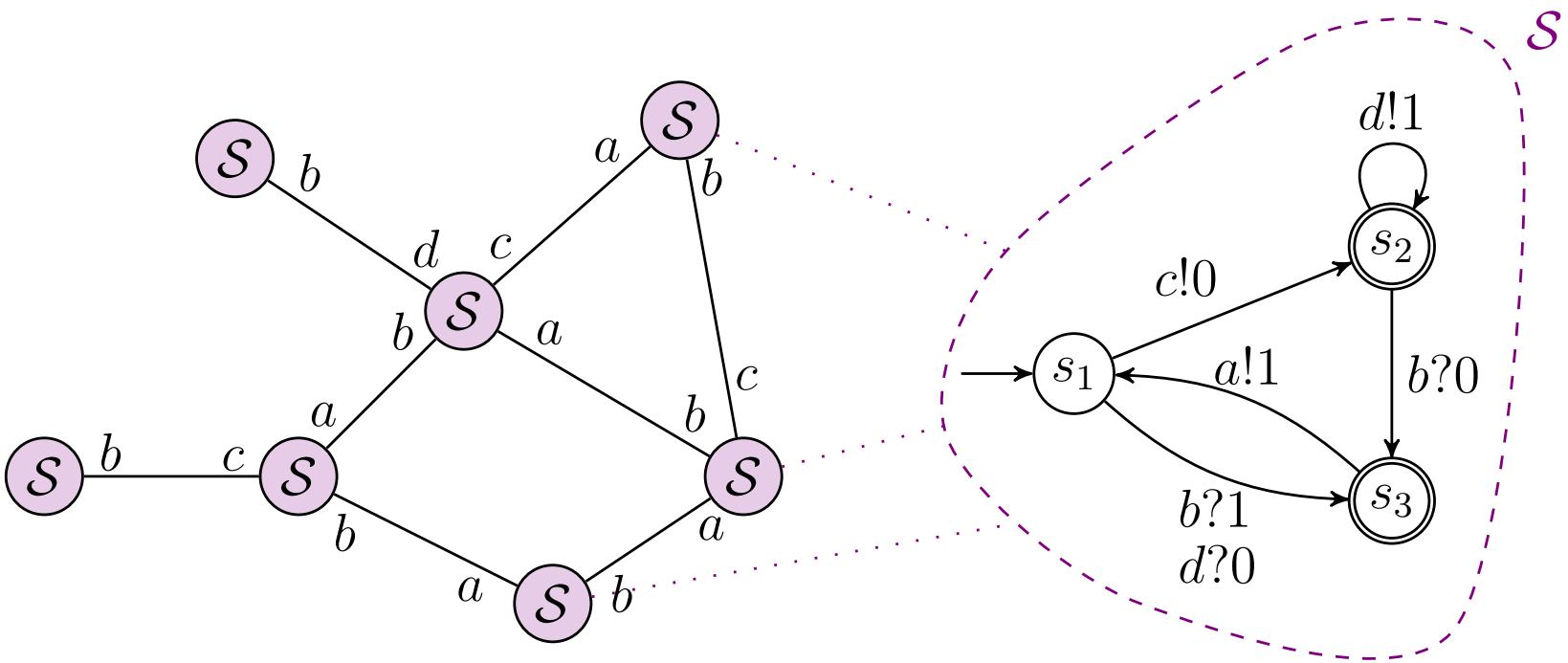
[B. Bollig 2014]



- Arbitrary number of identical processes
- Communication via FIFO channels
- Arbitrary topology of bounded degree

Parameterized Communicating Automata

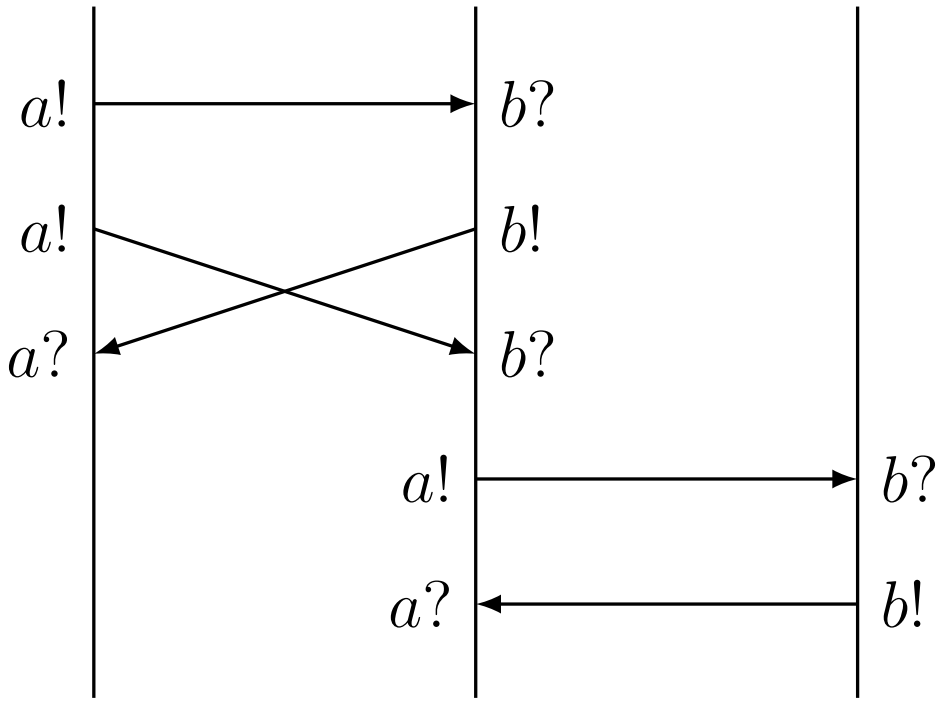
[B. Bollig 2014]



- Arbitrary number of identical processes
- Communication via FIFO channels
- Arbitrary topology of bounded degree

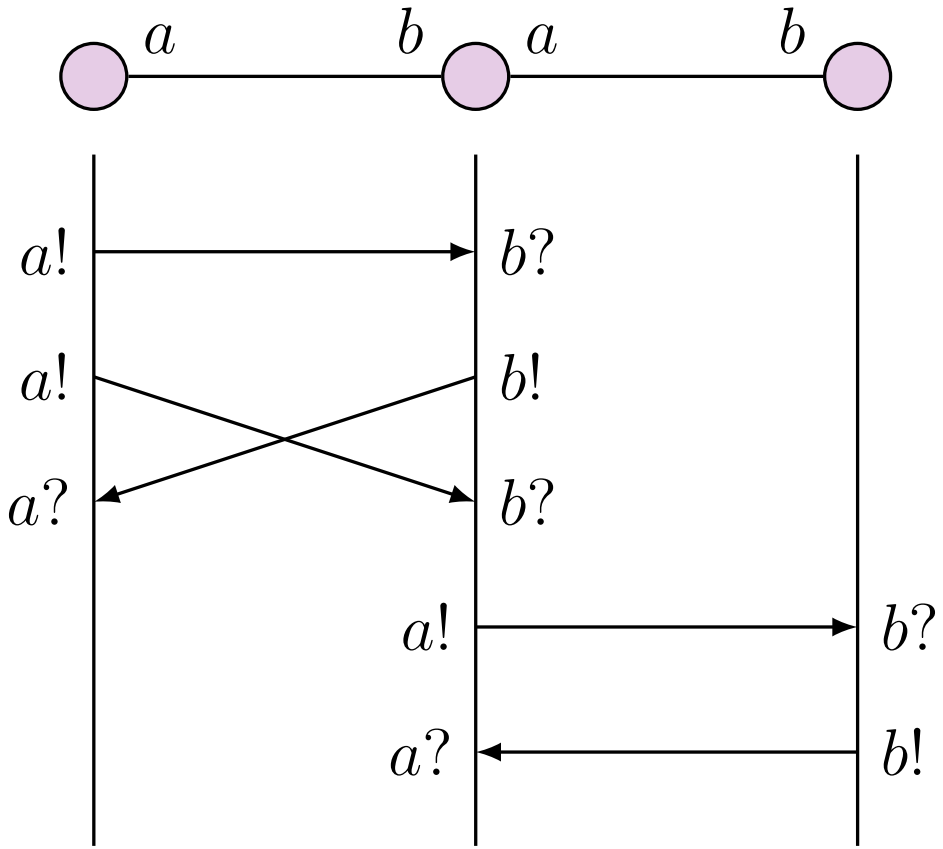
Semantics : Message Sequence Charts (MSC)

Behaviors as **graphs**:



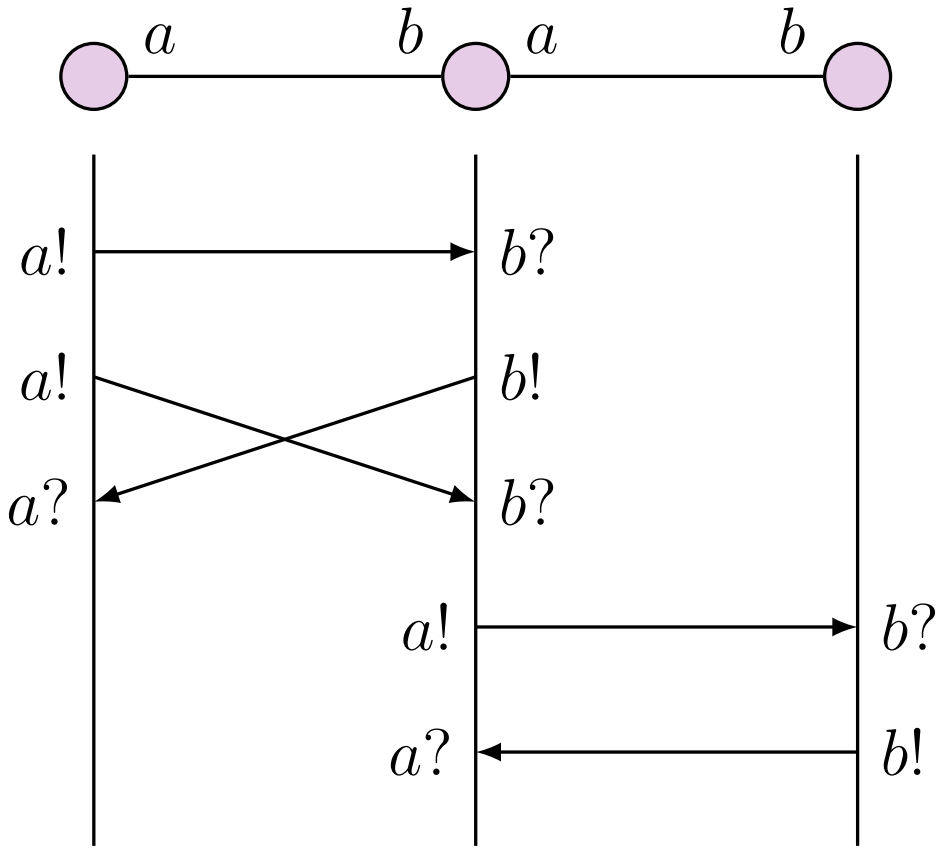
Semantics : Message Sequence Charts (MSC)

Behaviors as **graphs**:



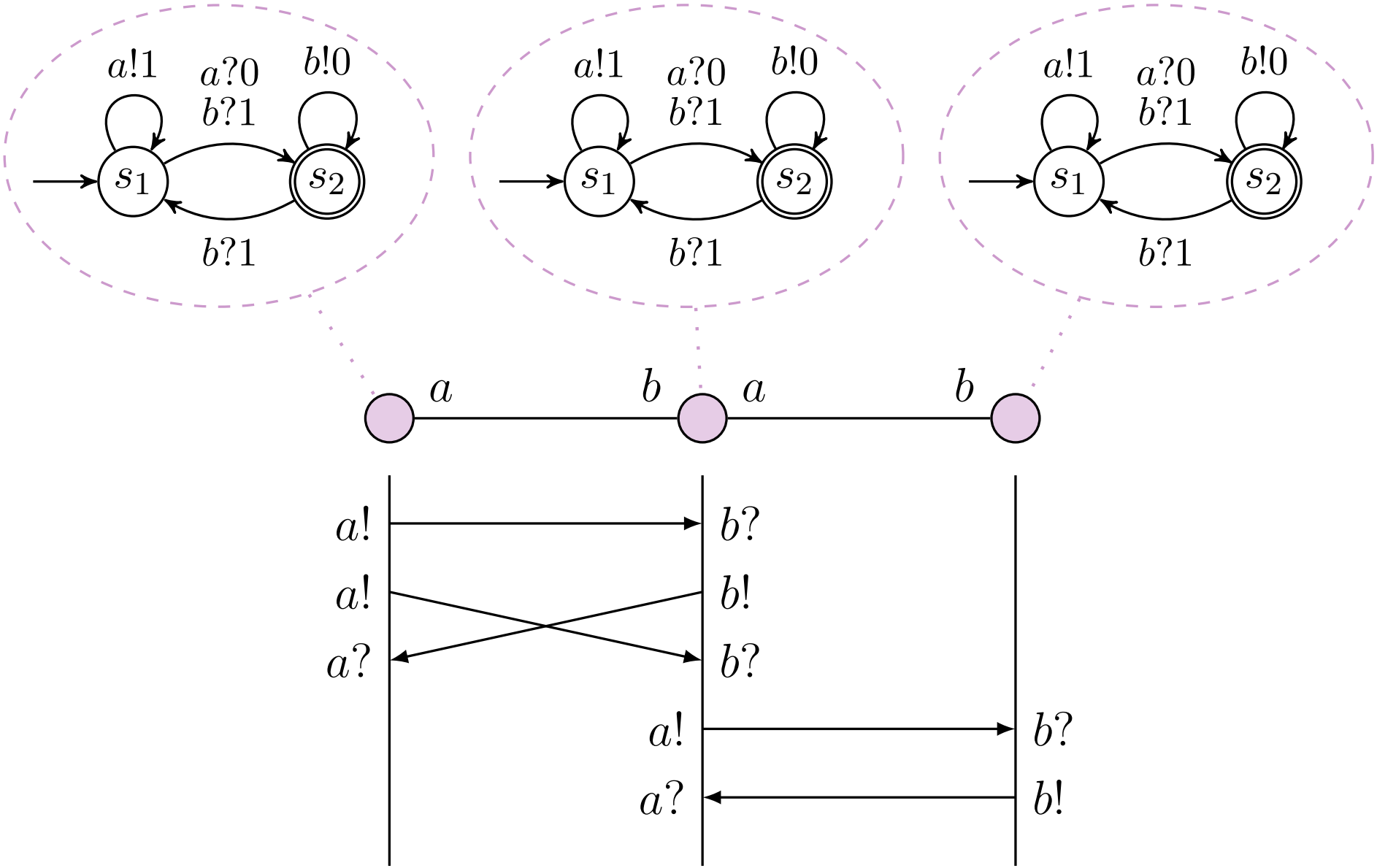
Semantics : Message Sequence Charts (MSC)

Behaviors as **graphs**:

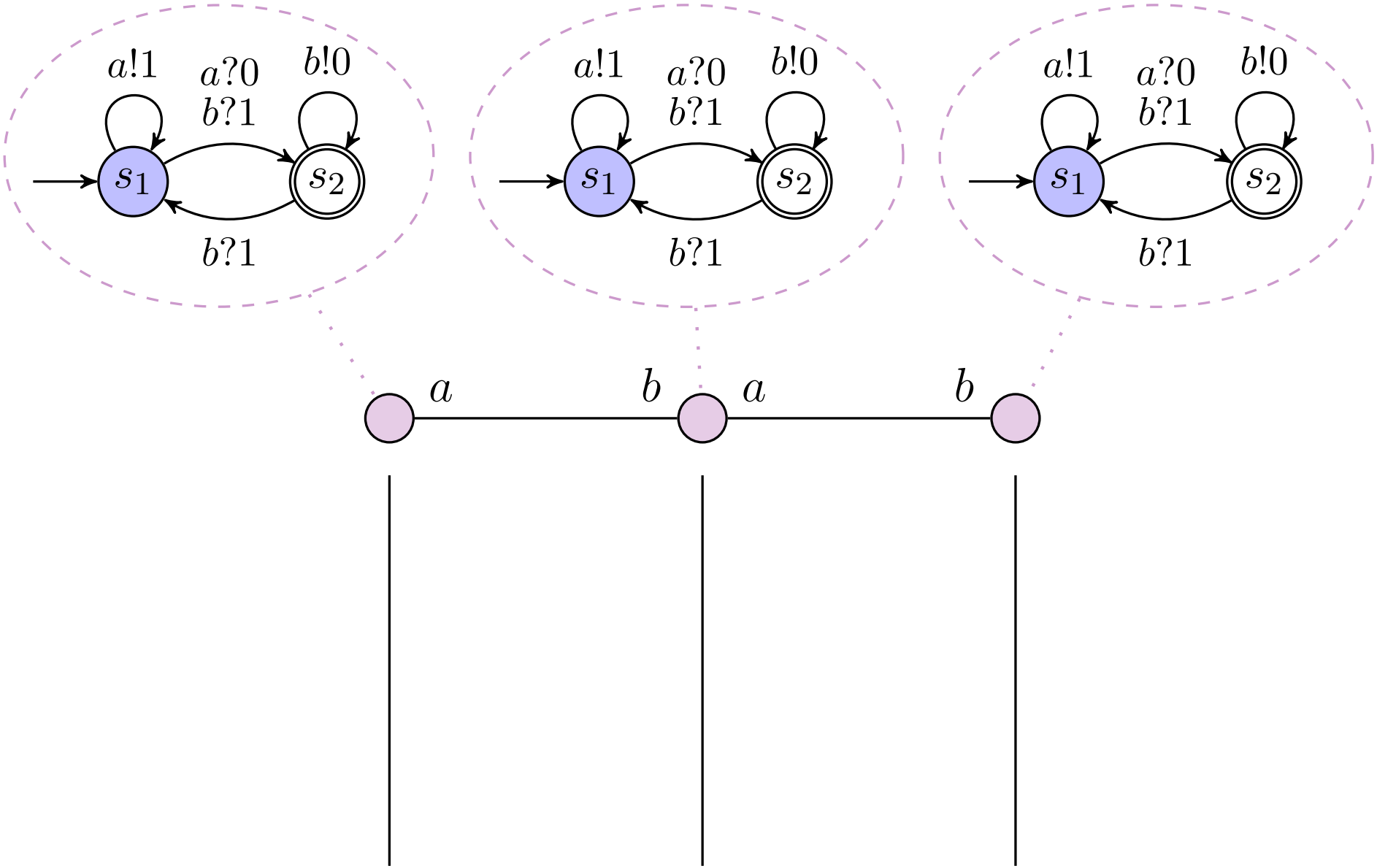


$L(\mathcal{S})$ is a set of MSCs on **arbirary topologies**.

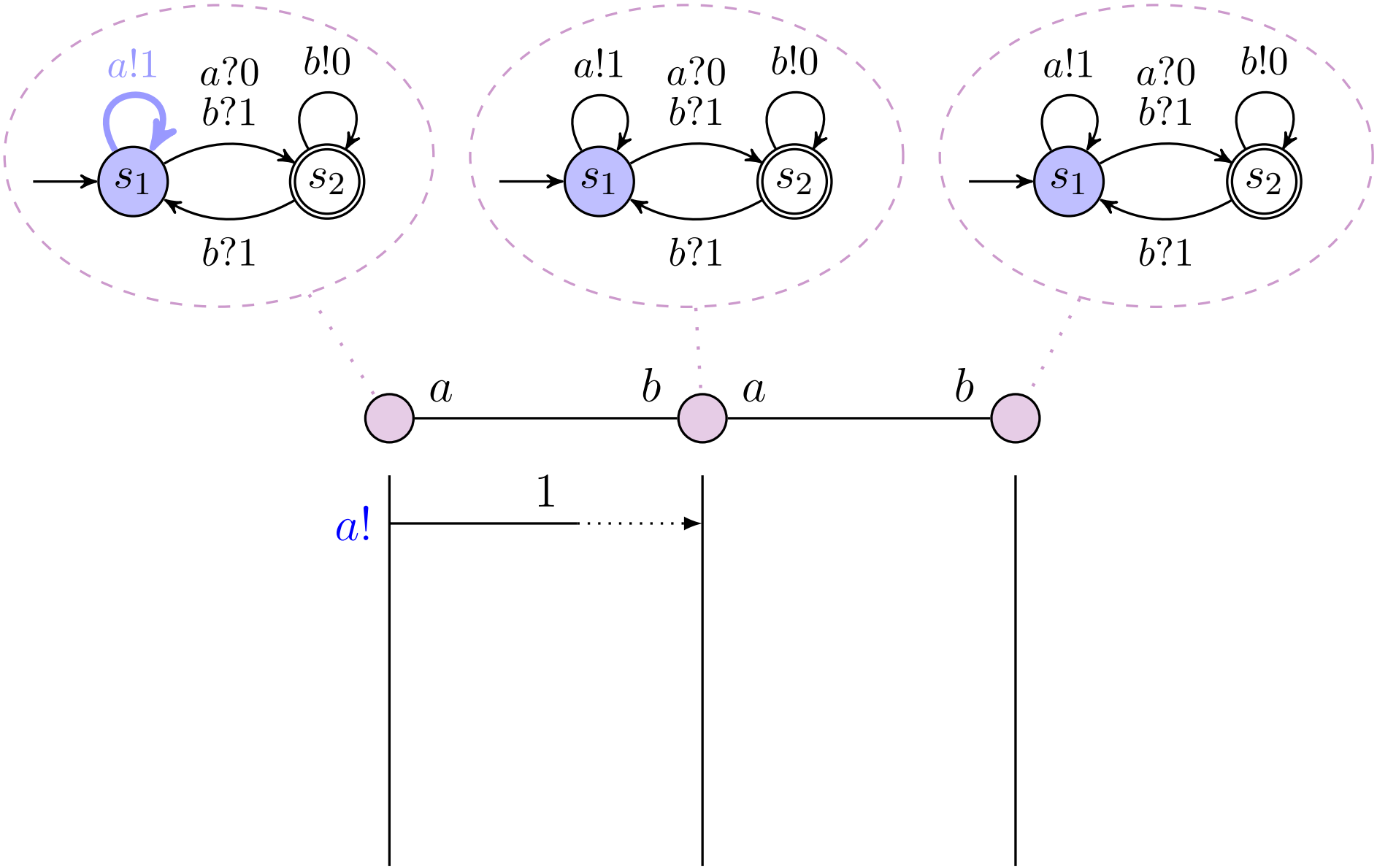
Example


a!1
a?0
b?1
b!0
s_1
s_2
b?1
a!1
a?0
b?1
b!0
s_1
s_2
b?1
a!1
a?0
b?1
b!0
s_1
s_2
b?1
a
b
a
b
a!
a!
a?
b?
b!
b?
a!
a?
b?
b!

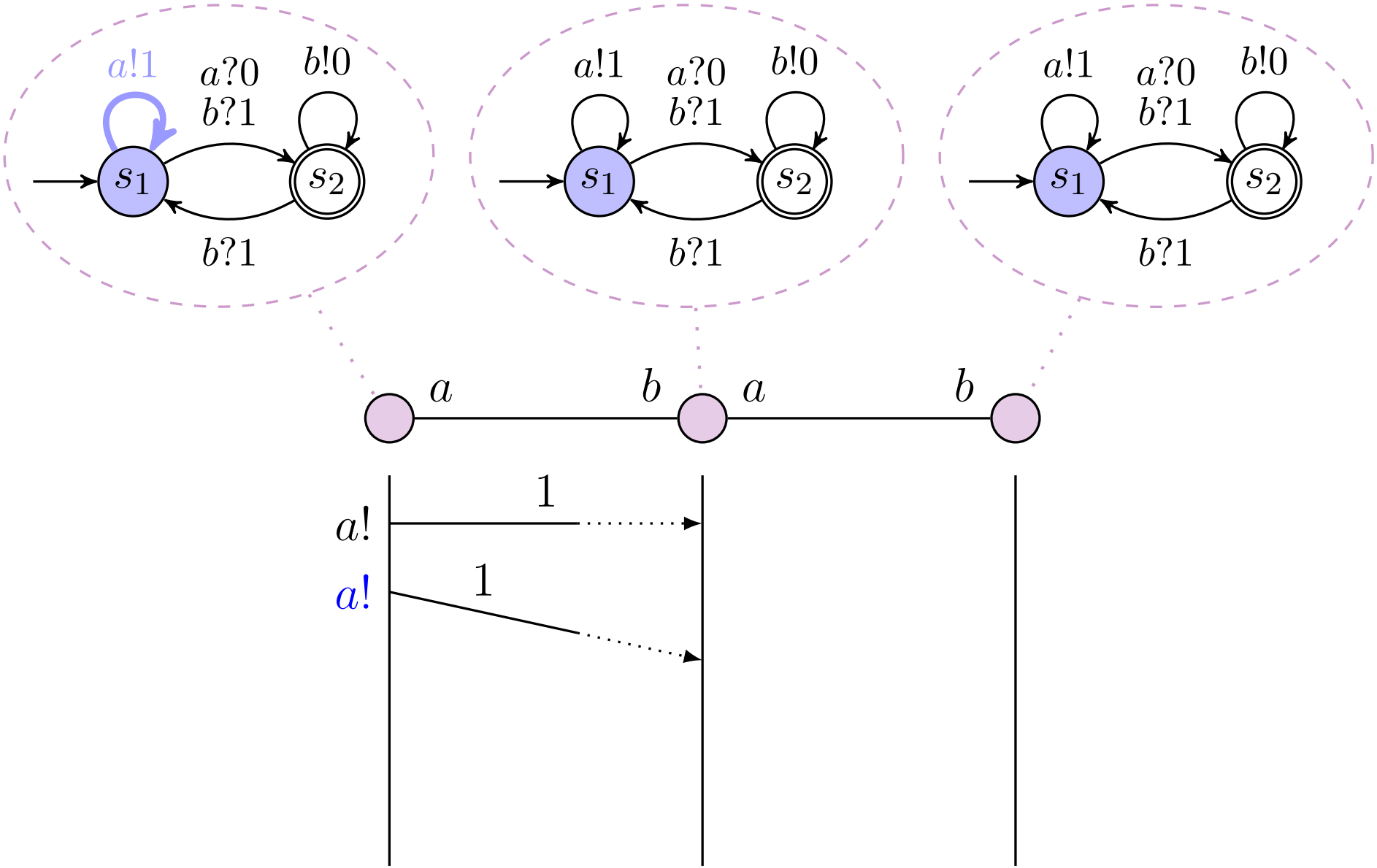
Example



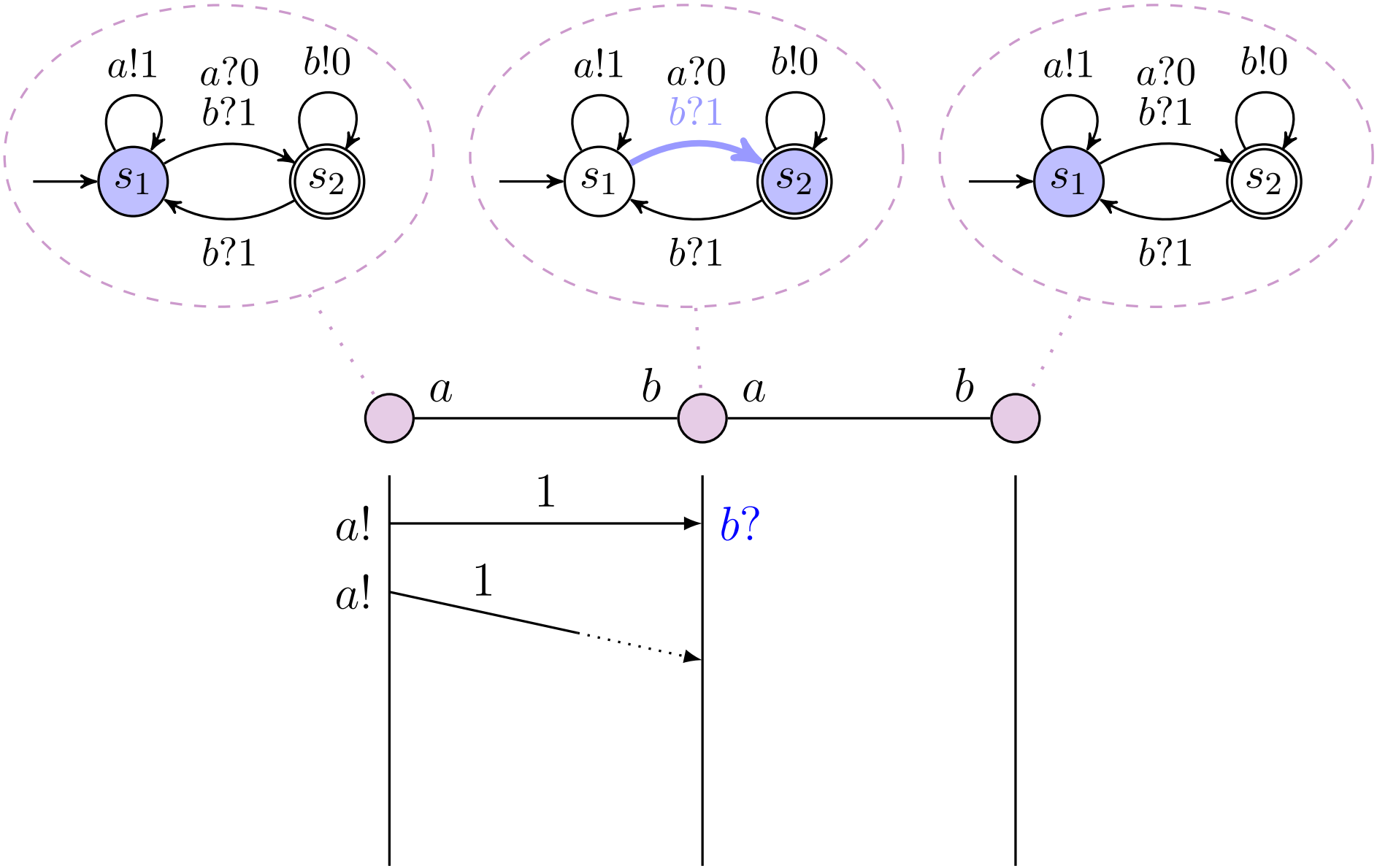
Example



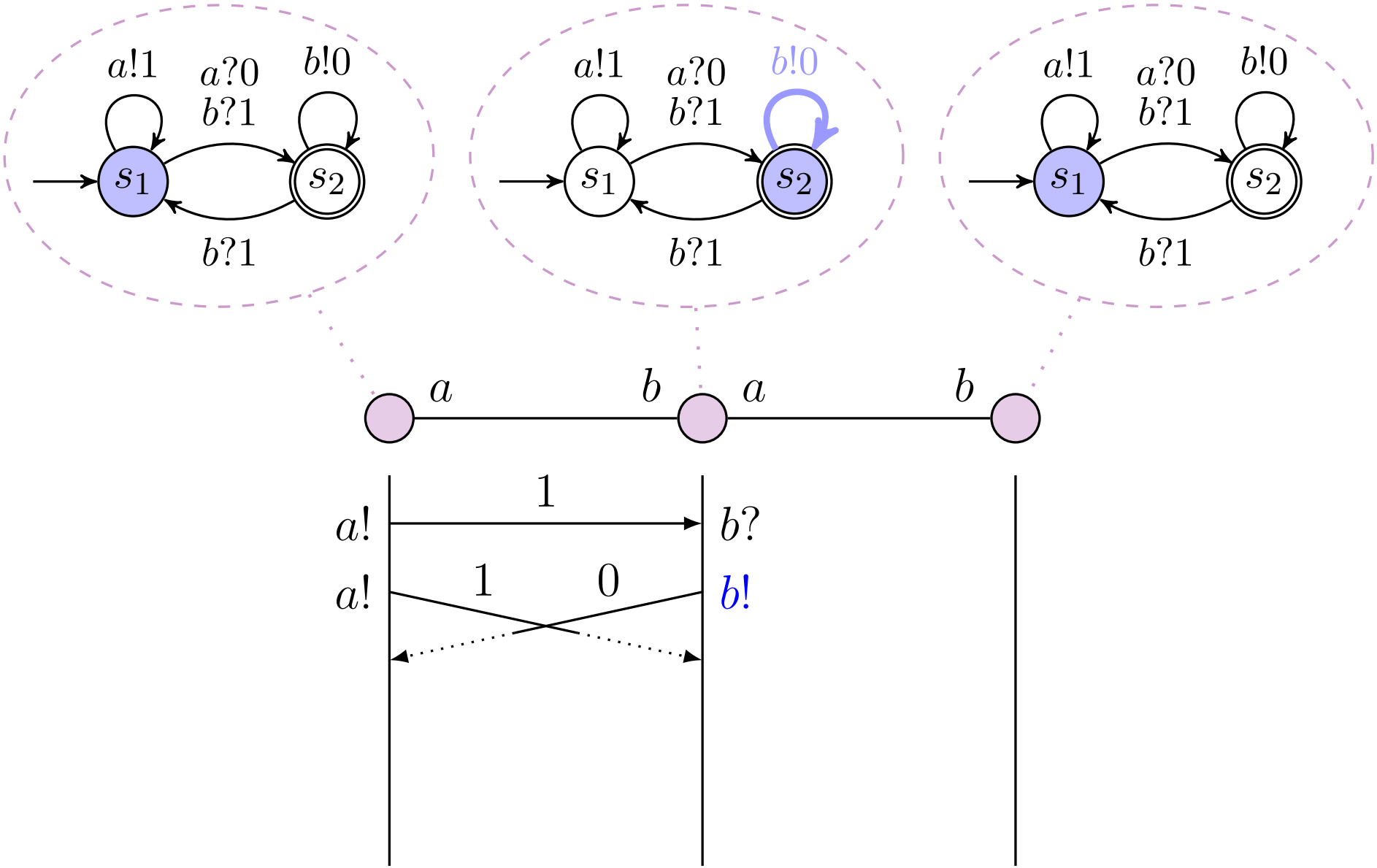
Example



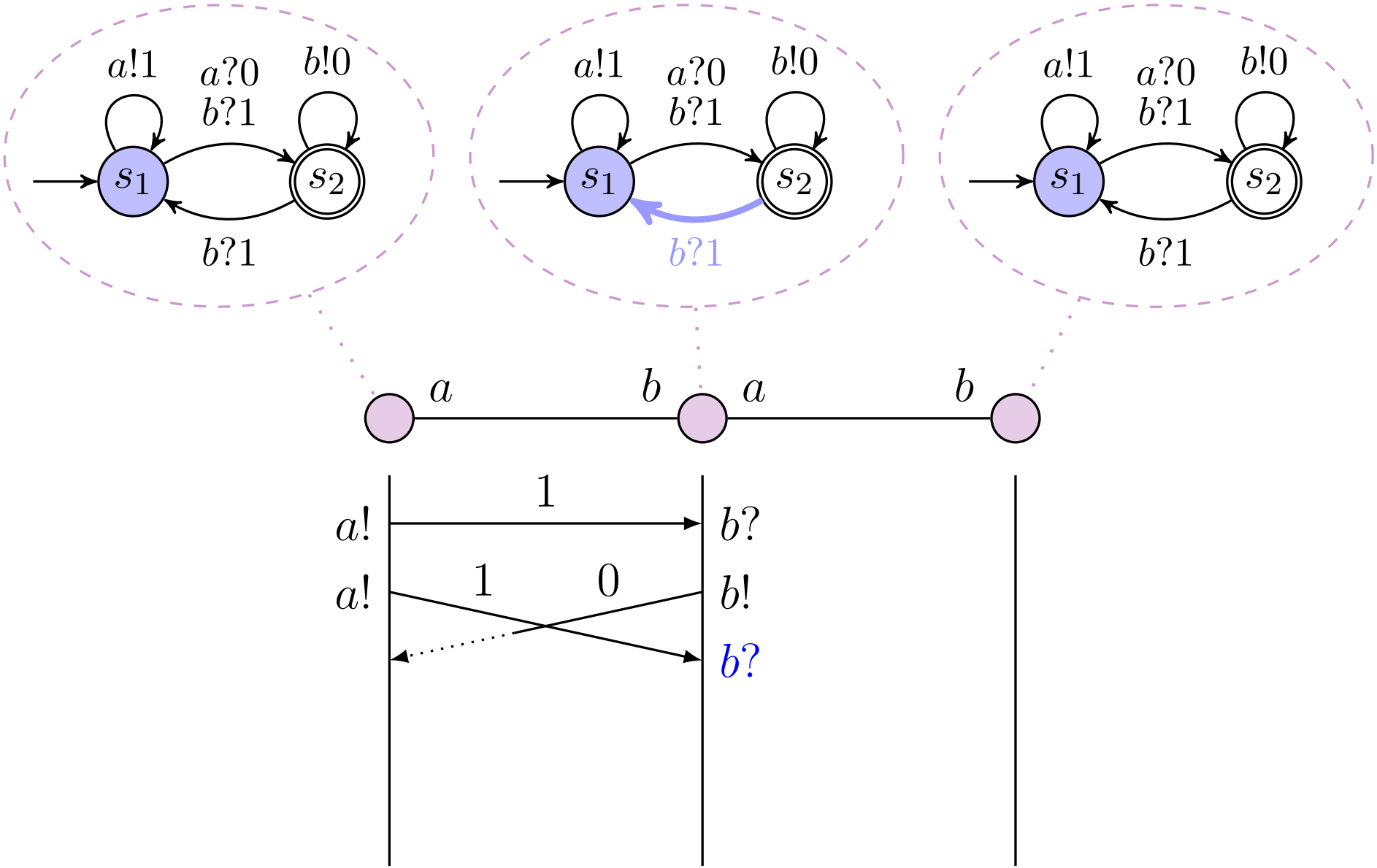
Example


a!1
a?0
b!0
b?1
s_1
s_2
b?1
a!1
a?0
b!0
b?1
s_1
s_2
b?1
a!1
a?0
b!0
b?1
s_1
s_2
b?1
a
b
a
b
a!
1
b?
a!
1

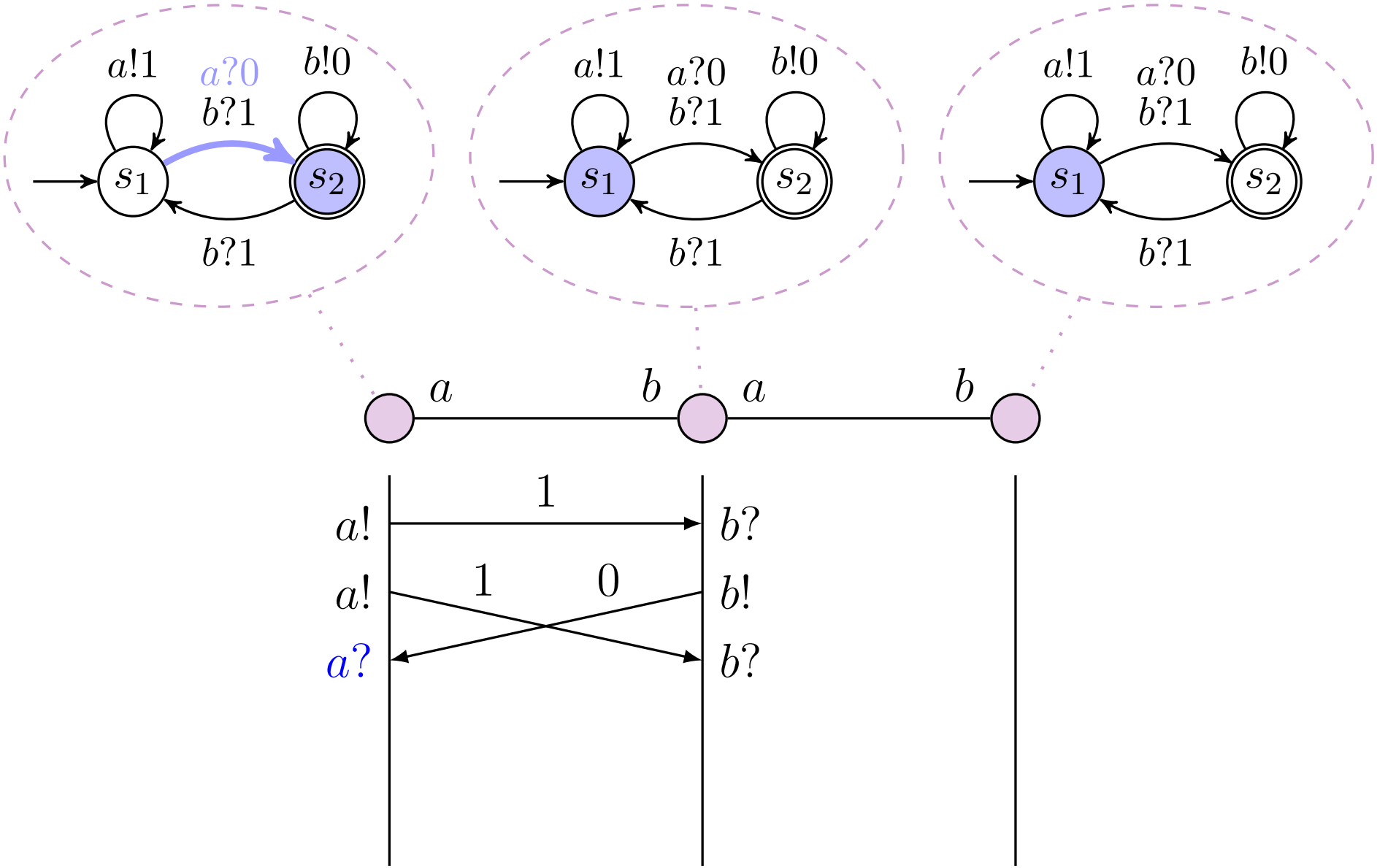
Example



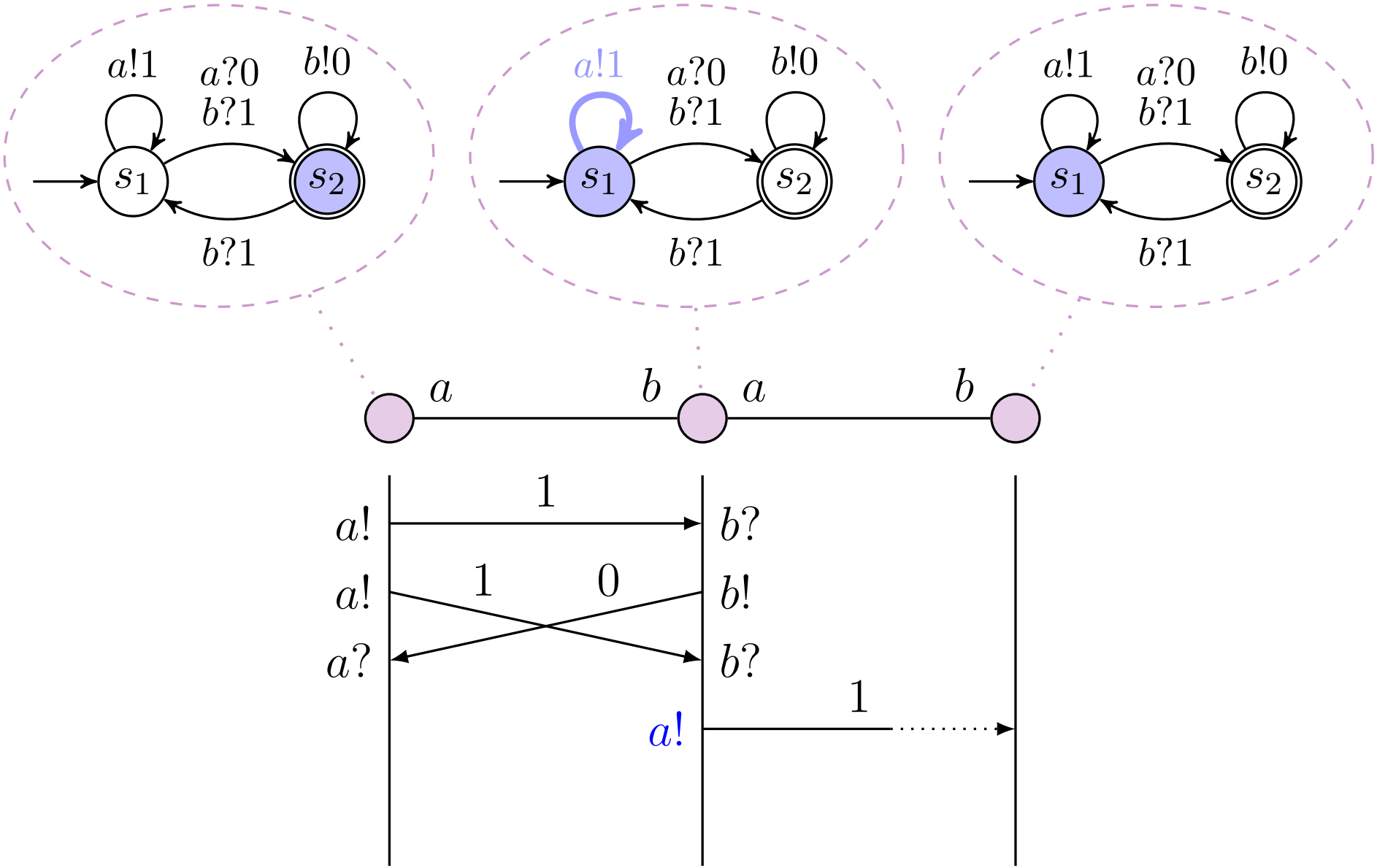
Example


a!1
a?0
b?1
b!0
s_1
s_2
b?1
a!1
a?0
b?1
b!0
s_1
s_2
b?1
a!1
a?0
b?1
b!0
s_1
s_2
b?1
a
b
a
b
a!
a!
1
1
0
b?
b!
b?

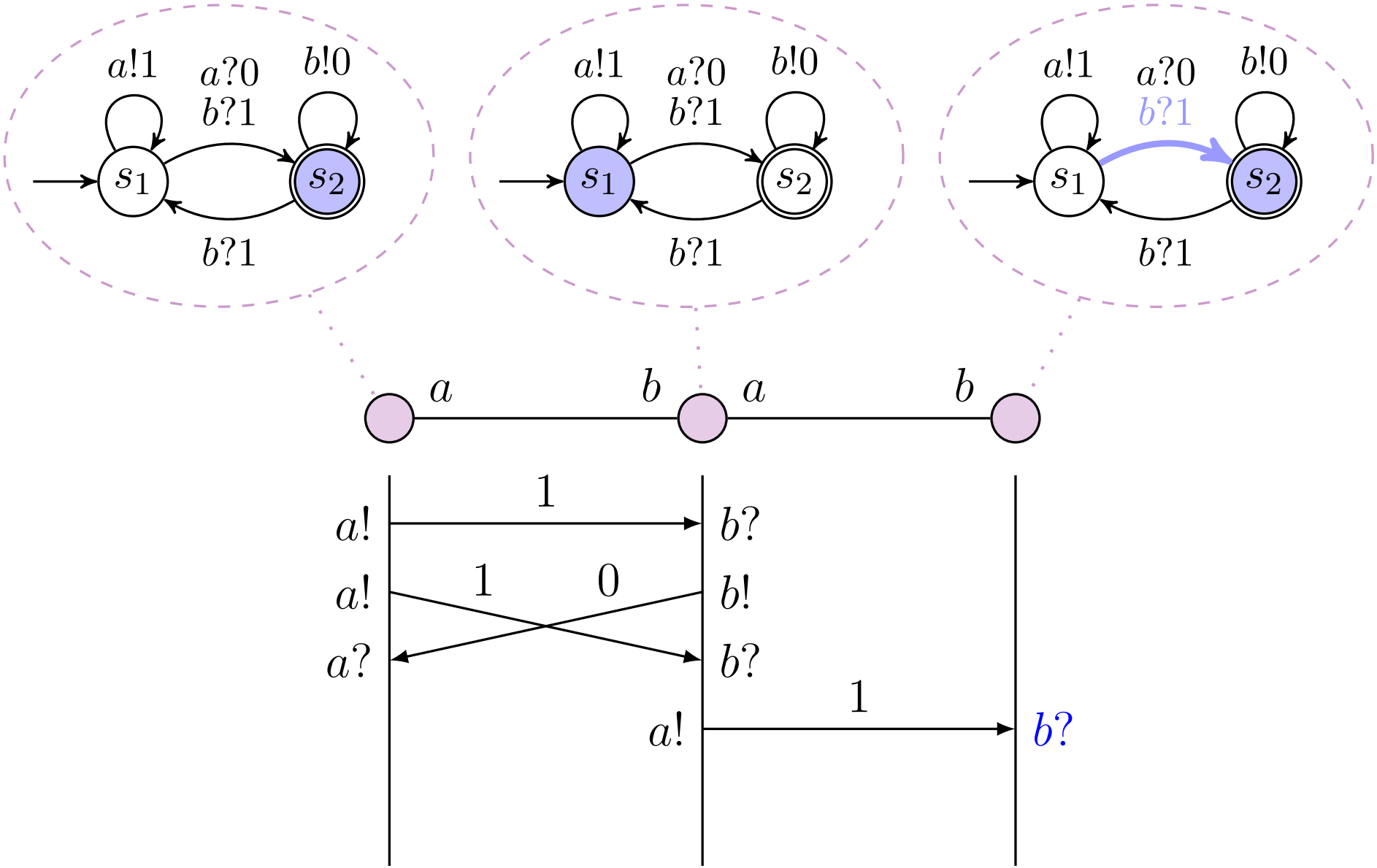
Example


a!1
a?0
b?1
b!0
s1
s2
b?1
a!1
a?0
b?1
b!0
s1
s2
b?1
a!1
a?0
b?1
b!0
s1
s2
b?1
a
b
a
b
1
a!
b?
a!
1
0
b!
a?
b?

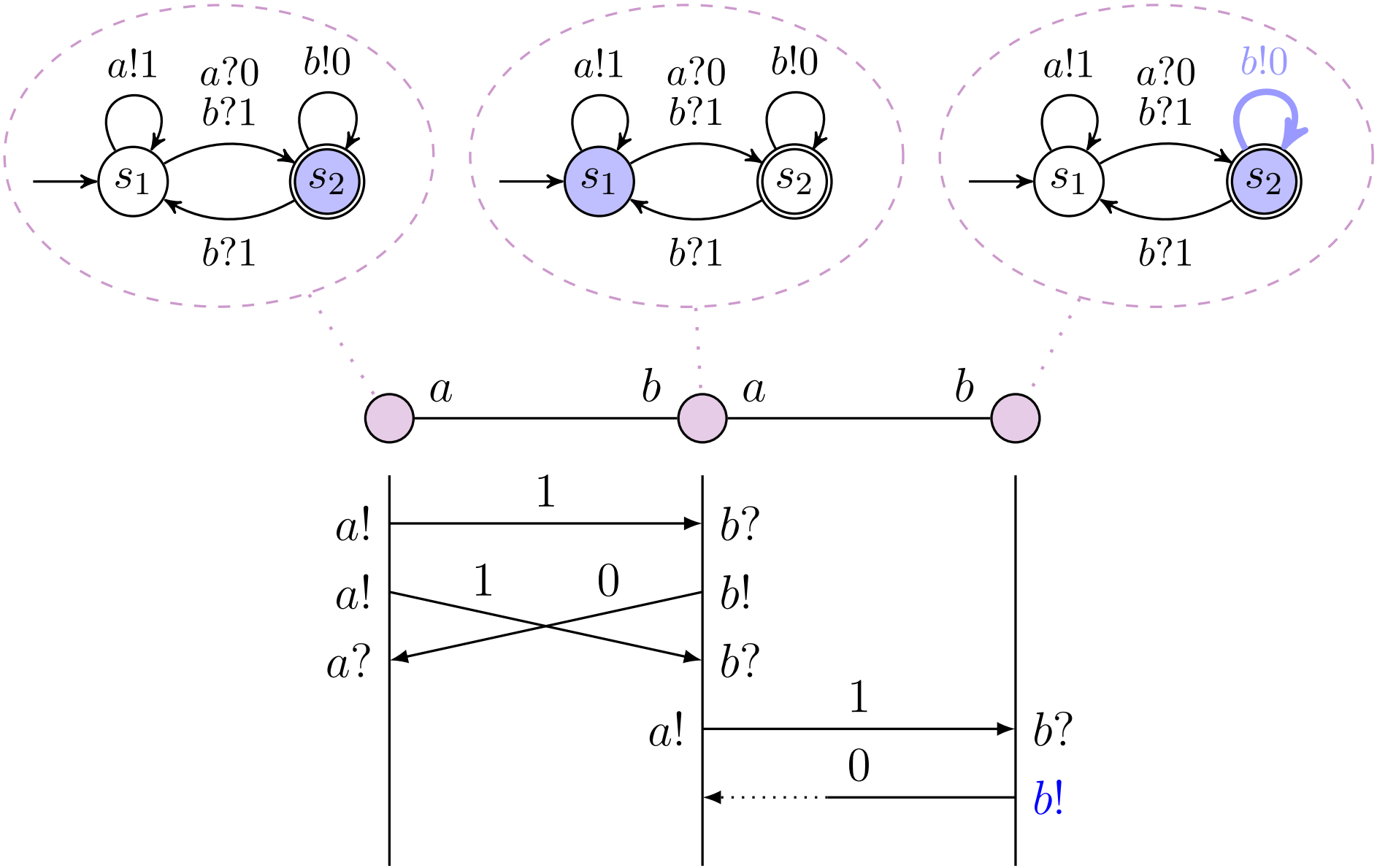
Example


a!1
a?0
b?1
b!0
s_1
s_2
b?1
a!1
a?0
b?1
b!0
s_1
s_2
b?1
a!1
a?0
b?1
b!0
s_1
s_2
b?1
a
b
a
b
1
a!
b?
a!
1
0
b!
a?
b?
1
a!

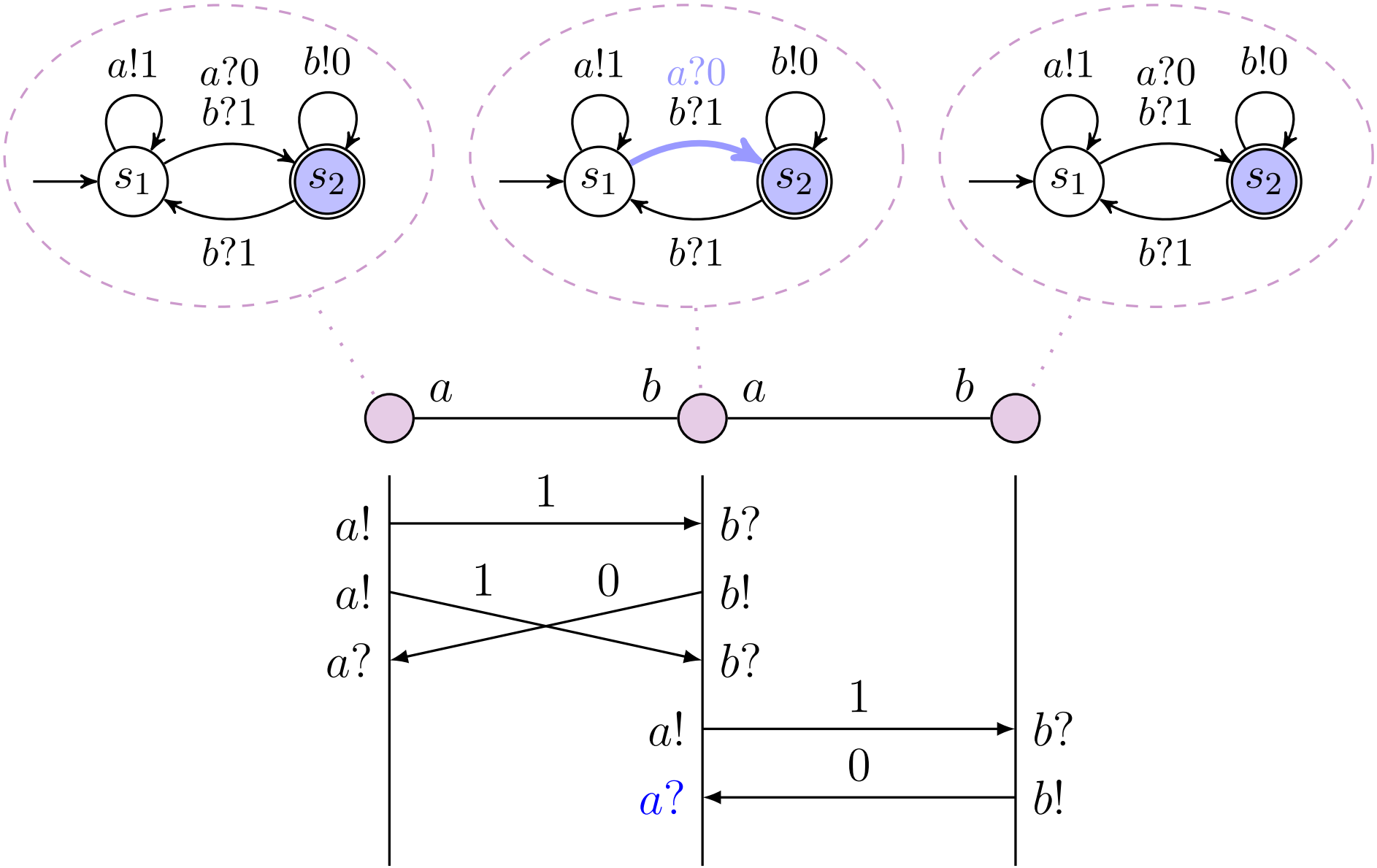
Example


a!1
a?0
b?1
b!0
s_1
s_2
b?1
a!1
a?0
b?1
b!0
s_1
s_2
b?1
a!1
a?0
b?1
b!0
s_1
s_2
b?1
a
b
a
b
1
a!
b?
a!
1
0
b!
a?
b?
1
a!
b?

Example



Example



Parameterized verification

Model-checking (fixed topology)

INPUT: $\mathcal{S}$, topology $\mathcal{T}$, specification φ
QUESTION: $\forall M \in L_{\mathcal{T}}(\mathcal{S}),\ M \models \varphi$?

Parameterized verification

Model-checking (fixed topology)

INPUT: $\mathcal{S}$, topology $\mathcal{T}$, specification φ
QUESTION: $\forall M \in L_{\mathcal{T}}(\mathcal{S}),\ M \models \varphi$?

Model-checking (parameterized case)

INPUT: $\mathcal{S}$, specification φ
QUESTION: $\forall \mathcal{T},\ \forall M \in L_{\mathcal{T}}(\mathcal{S}),\ M \models \varphi$?

Parameterized verification

Model-checking (fixed topology)

INPUT: $\mathcal{S}$, topology $\mathcal{T}$, specification φ
QUESTION: $\forall M \in L_{\mathcal{T}}(\mathcal{S}),\ M \models \varphi$?

Model-checking (parameterized case)

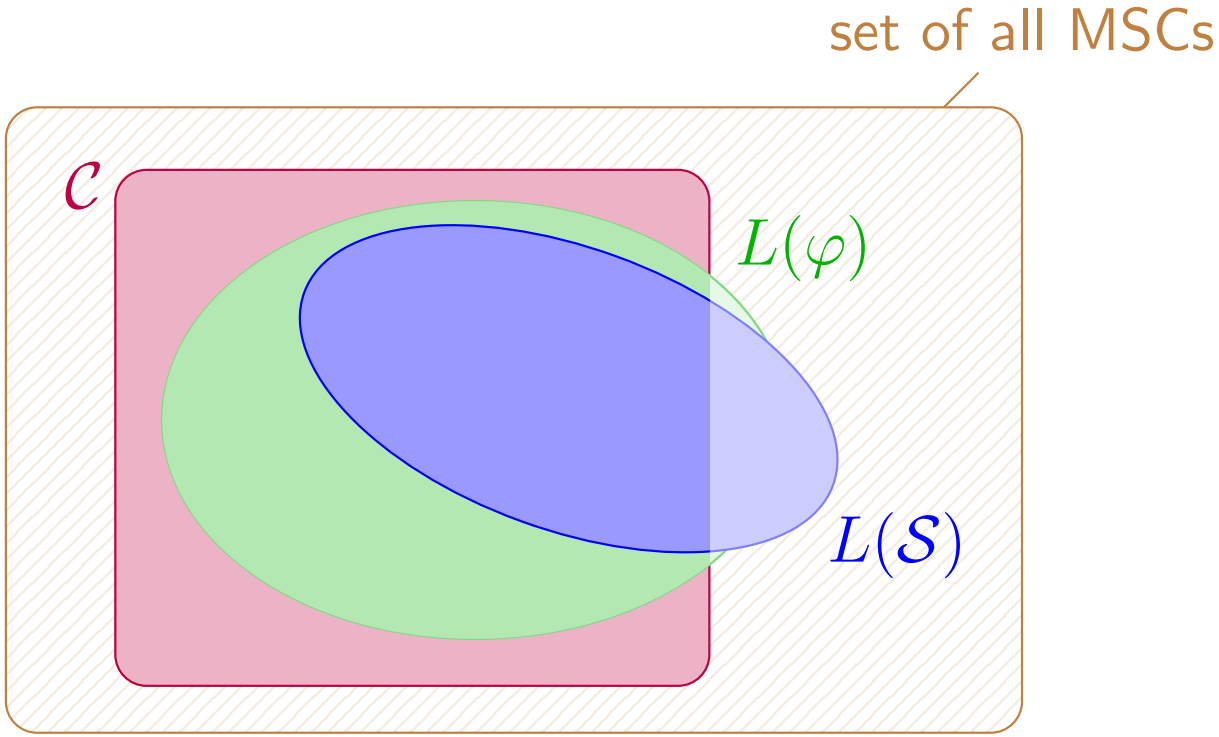
INPUT: $\mathcal{S}$, specification φ
QUESTION: $\forall \mathcal{T},\ \forall M \in L_{\mathcal{T}}(\mathcal{S}),\ M \models \varphi$?

Both are **undecidable**

Undecidability

Topology:	**Fixed**	**Parameterized, arbitrary**	**Parameterized, pipelines**
General case	✗	✗	✗
Bounded number of actions on each process	✓	✗	✓
Synchronous communication	✓	✗	✗

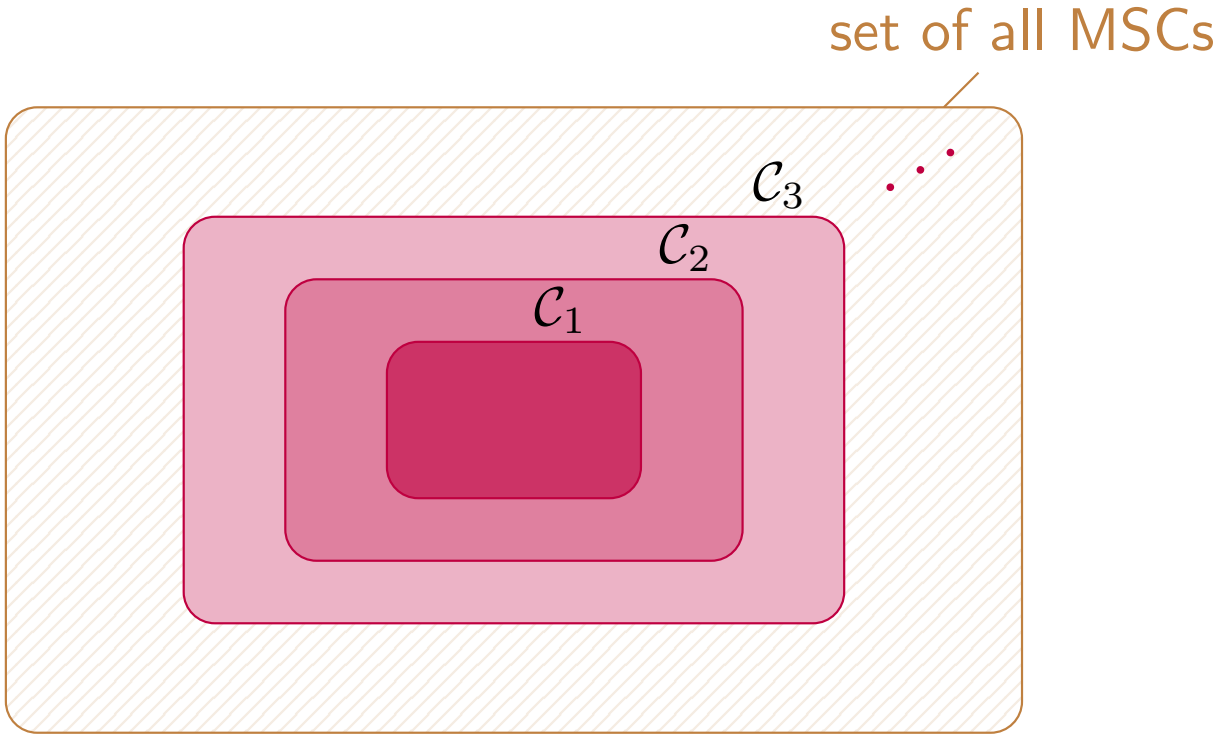
Underapproximate verification



Model-Checking: $L(\mathcal{S}) \subseteq L(\varphi)?$

$\mathcal{C}$-Model-Checking: $\mathcal{C} \cap L(\mathcal{S}) \subseteq L(\varphi)?$

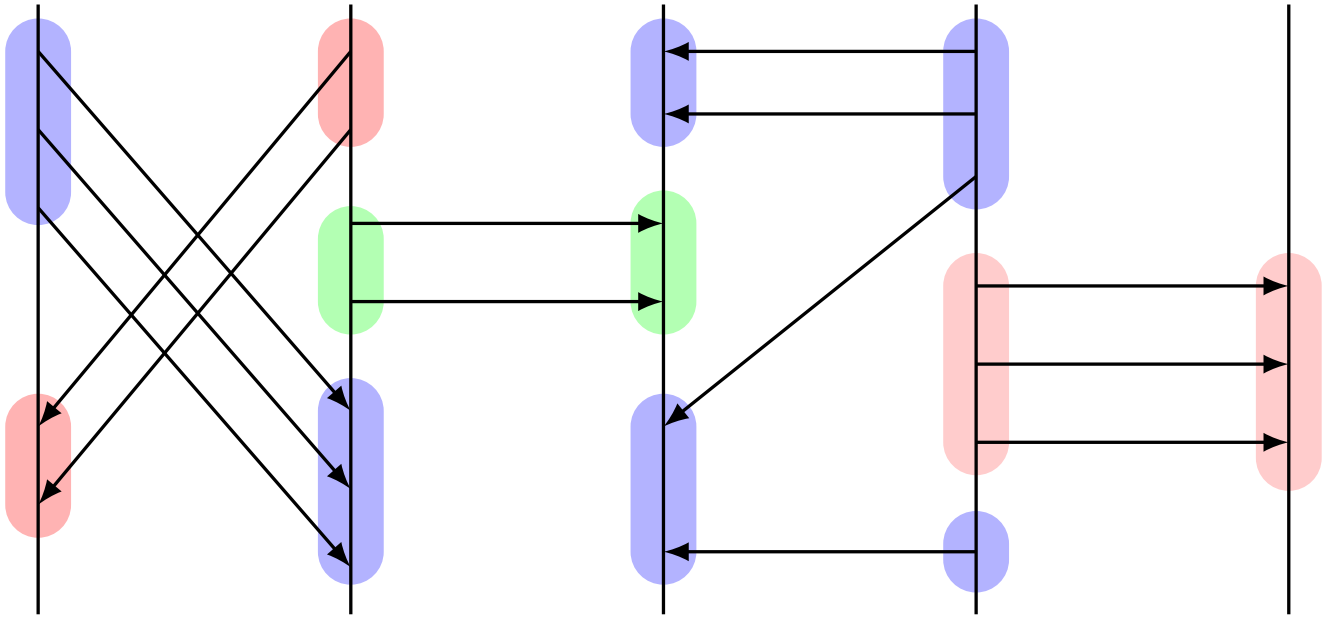
Underapproximate verification



Model-Checking: $L(\mathcal{S}) \subseteq L(\varphi)?$

$\mathcal{C}$-Model-Checking: $\mathcal{C}_k \cap L(\mathcal{S}) \subseteq L(\varphi)?$

Context-bounded MSCs



A 3-context-bounded MSC

Decidability results

Fixed topology: many decidable underapproximation classes, including:

- k-context-bounded MSCs
- k-existentially-bounded MSCs
- MSCs of tree-width/split-width at most k
- ...

Decidability results

Fixed topology: many decidable underapproximation classes, including:

- k-context-bounded MSCs
- k-existentially-bounded MSCs
- MSCs of tree-width/split-width at most k
- ...

Parameterized topology:

- Decidable for context-bounded MSCs over pipeline/ring/tree topologies, assuming synchronous communication.

Split-width

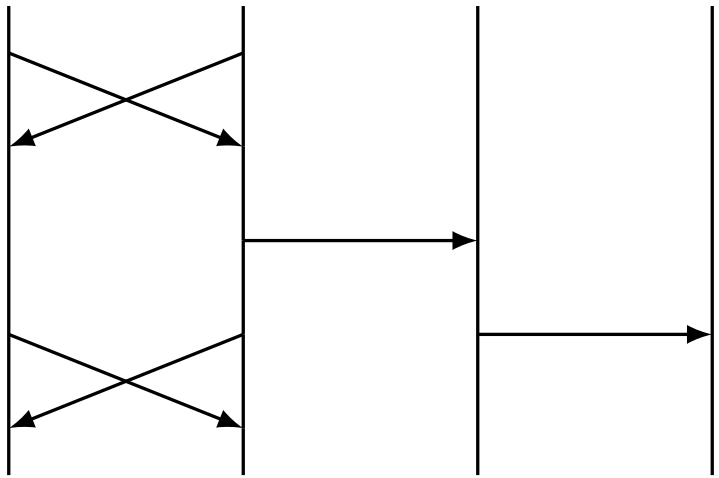
- Apply graph decomposition techniques (tree-width, clique-width...) to MSCs
- We work with split-width, equivalent but better suited for MSCs

Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

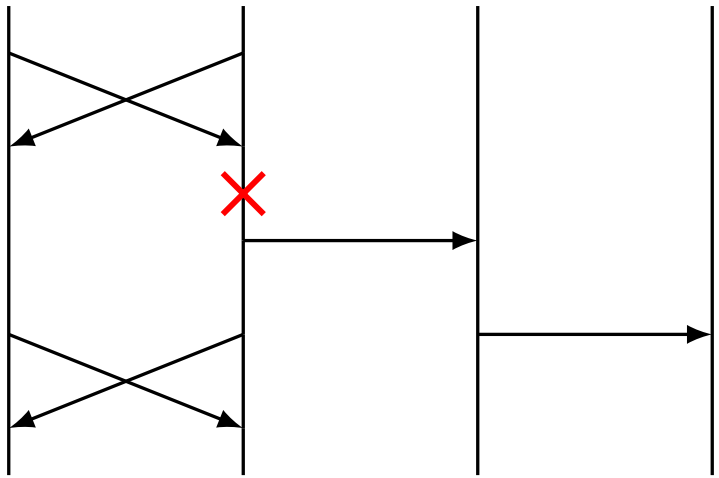


Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

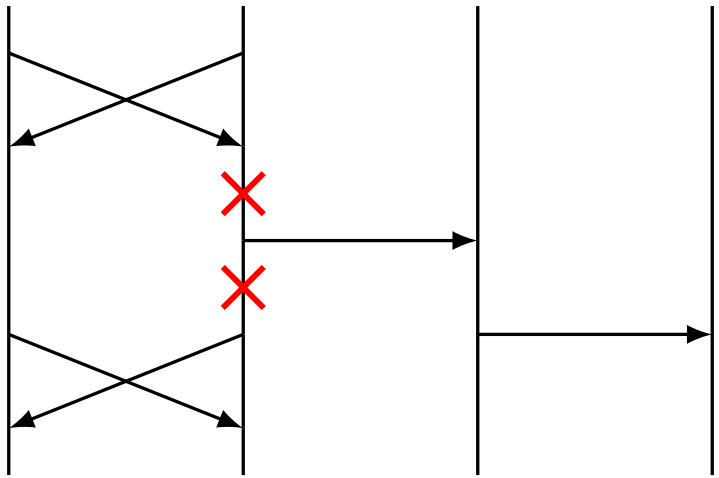


Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

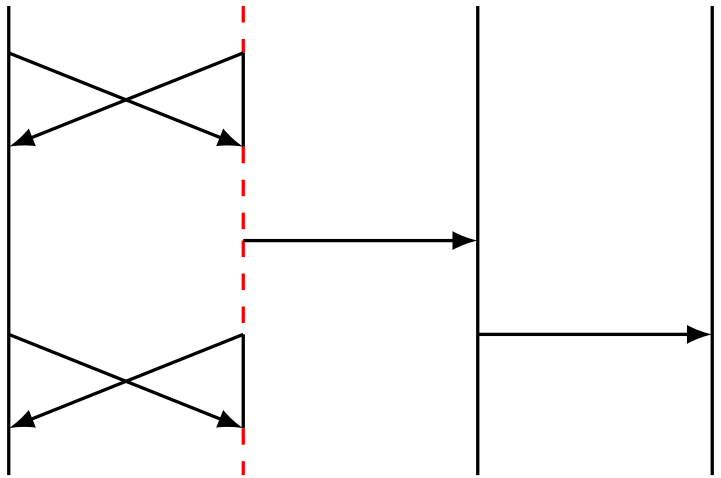


Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

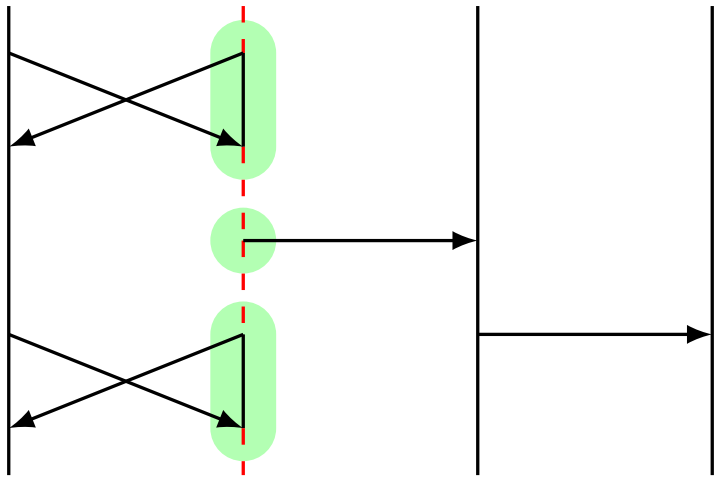


Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.



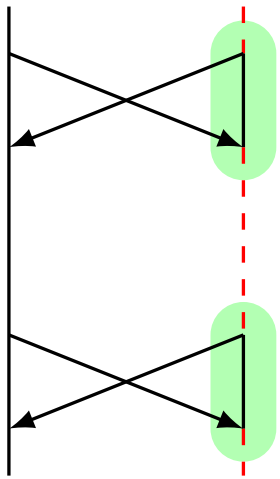
- 1 open process
- 3 blocks

Split-width

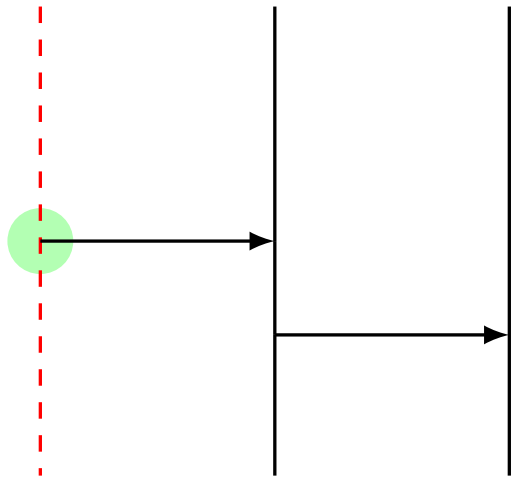
Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.



- 1 open process
- 2 blocks



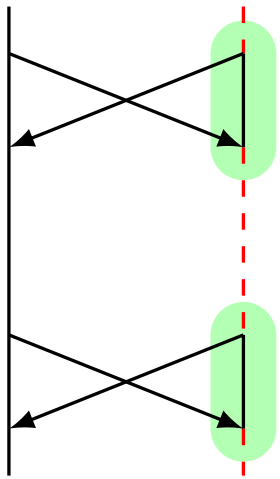
- 1 open process
- 1 block

Split-width

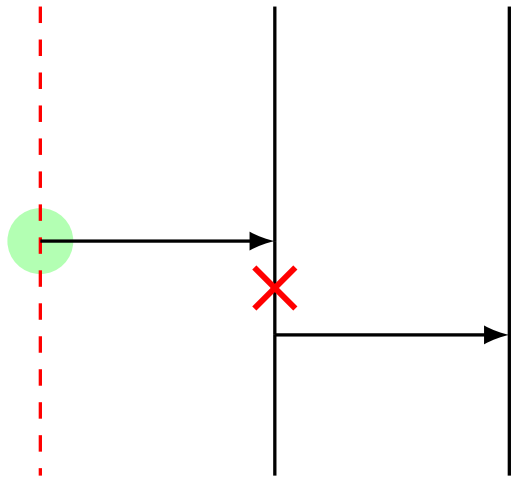
Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.



- 1 open process
- 2 blocks



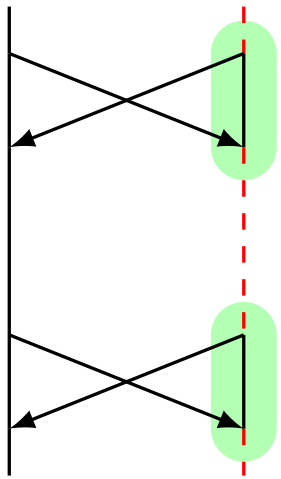
- 1 open process
- 1 block

Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

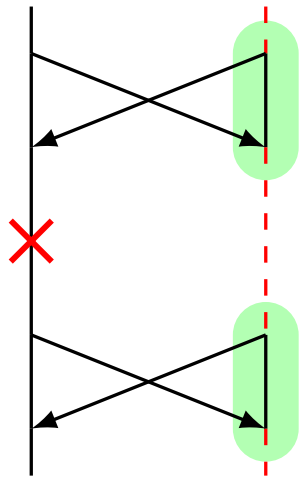


Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

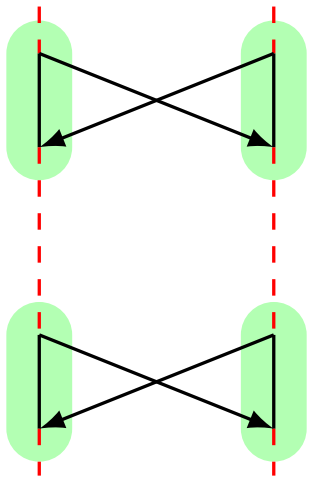


Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.



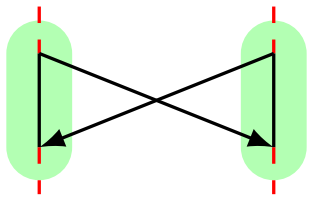
- 2 open processes
- 4 blocks

Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

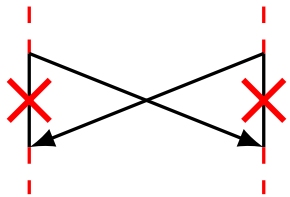


Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

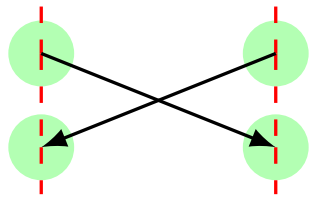


Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.



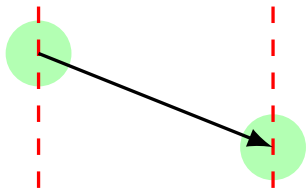
- 2 open processes
- 4 blocks

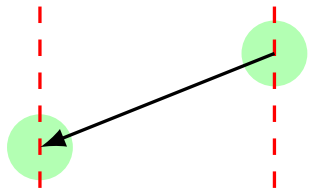
Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.





Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

$\longrightarrow$ Eve wins.

Split-width

Two-player game on M (with budget k) :

- Eve cuts process edges, without introducing more than k blocks
- Adam choses a connected component

Example : with budget 4.

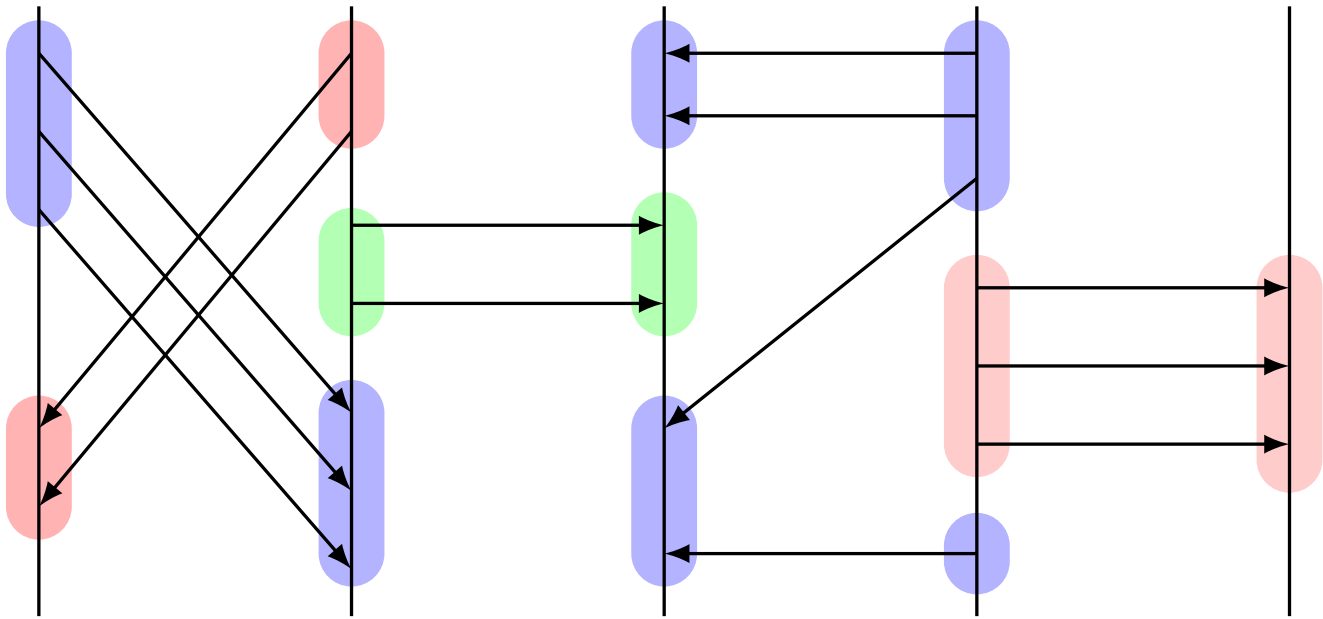
$\longrightarrow$ Eve wins.

split-width$(M) =$
least k for which Eve has a winning strategy

Example

Lemma

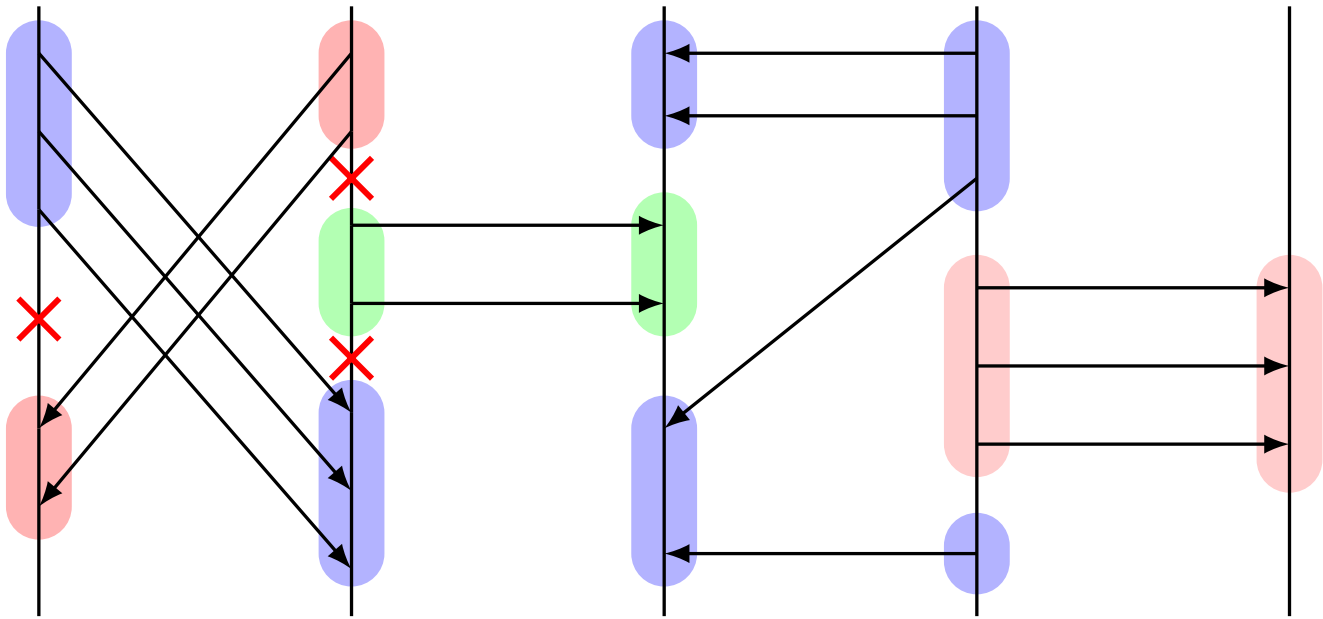
A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.



Example

Lemma

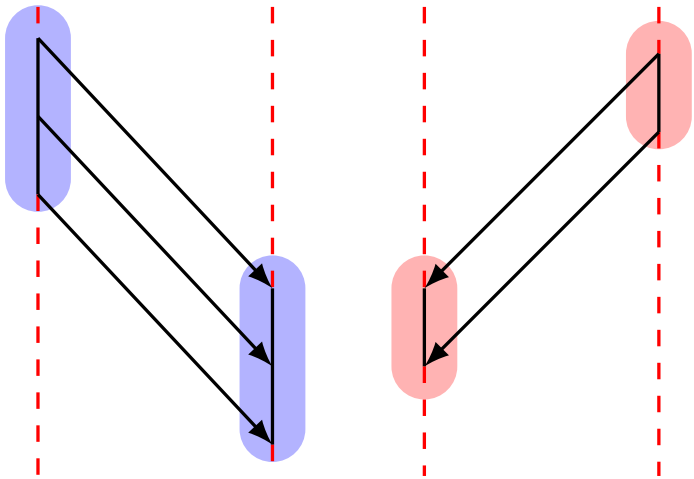
A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.

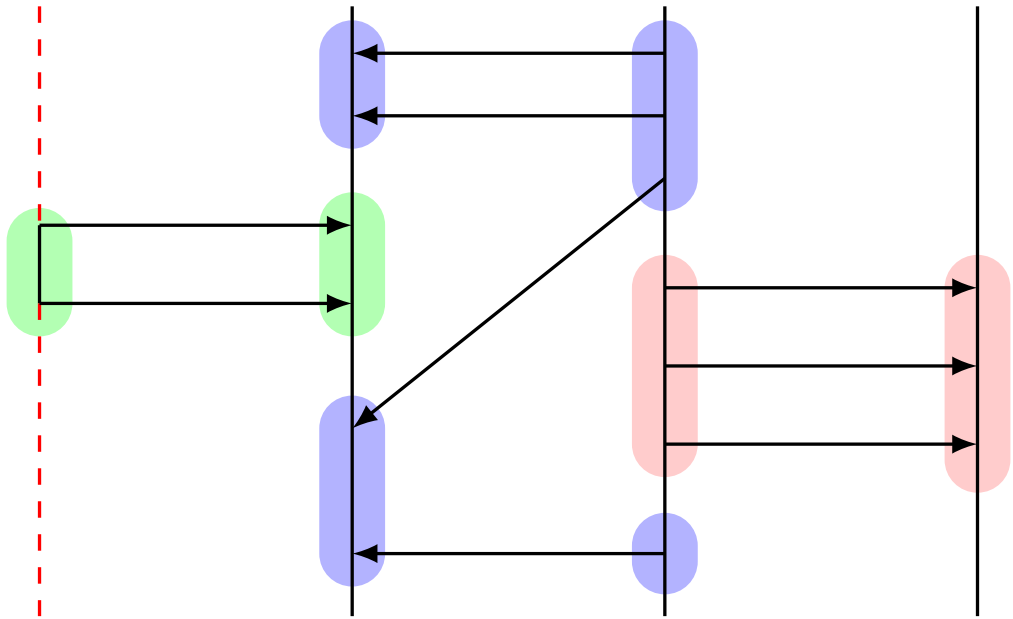


Example

Lemma

A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.

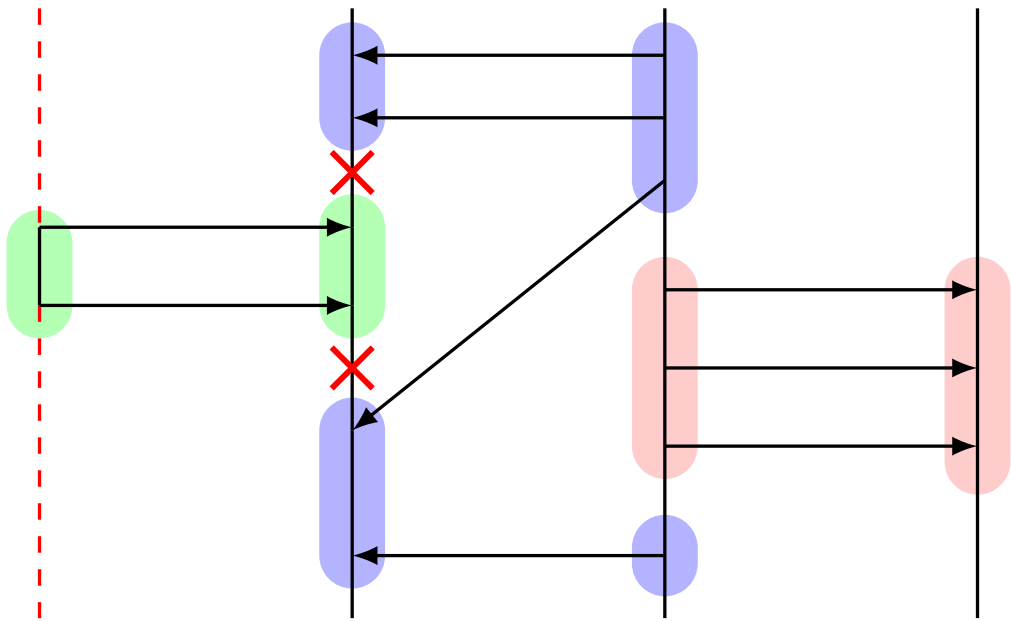




Example

Lemma

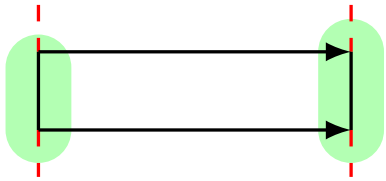
A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.

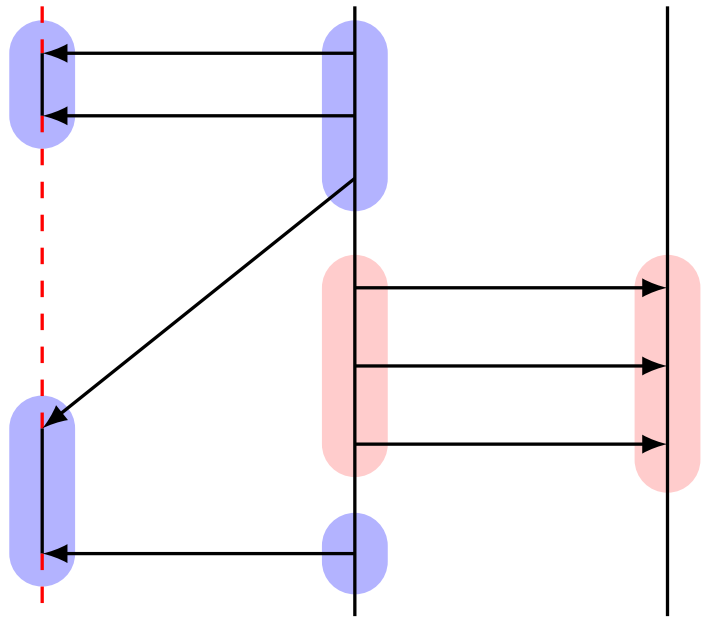


Example

Lemma

A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.

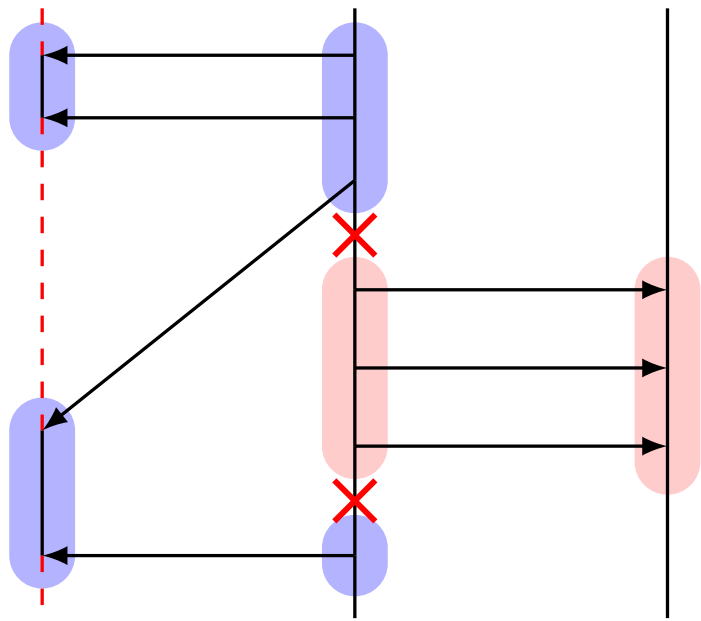




Example

Lemma

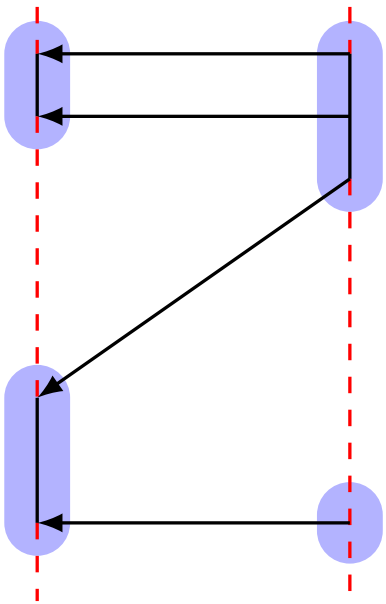
A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.

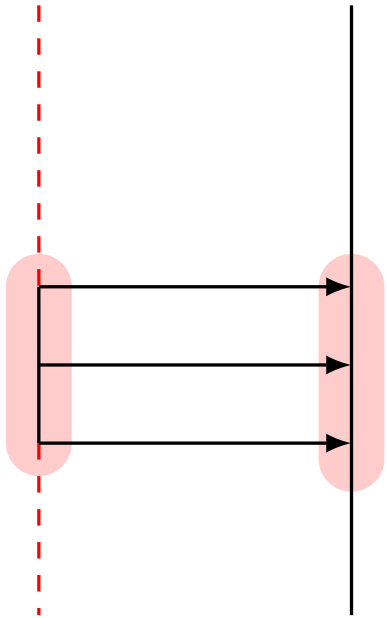


Example

Lemma

A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.

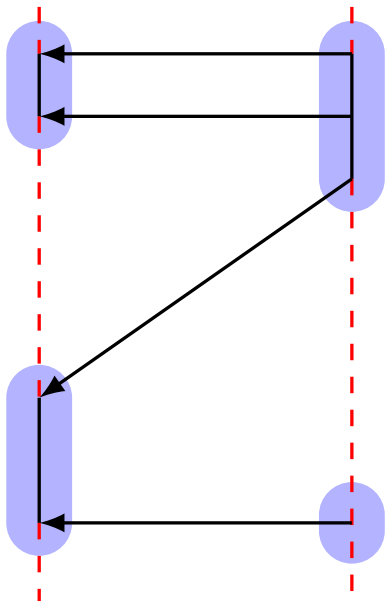




Example

Lemma

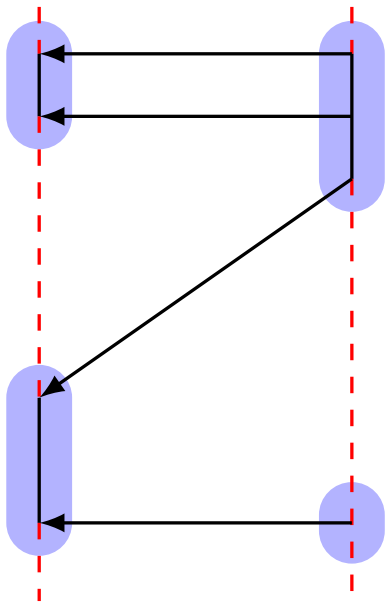
A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.



Example

Lemma

A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.

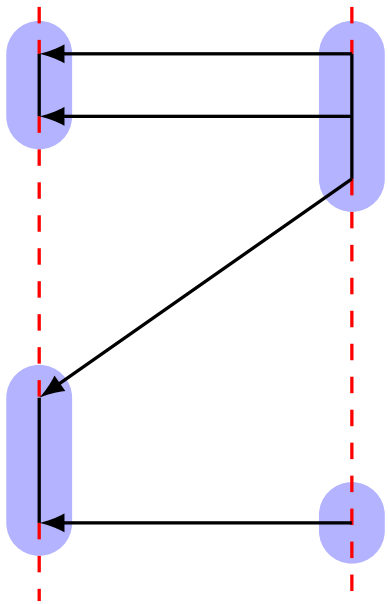


- At most k blocks on each process

Example

Lemma

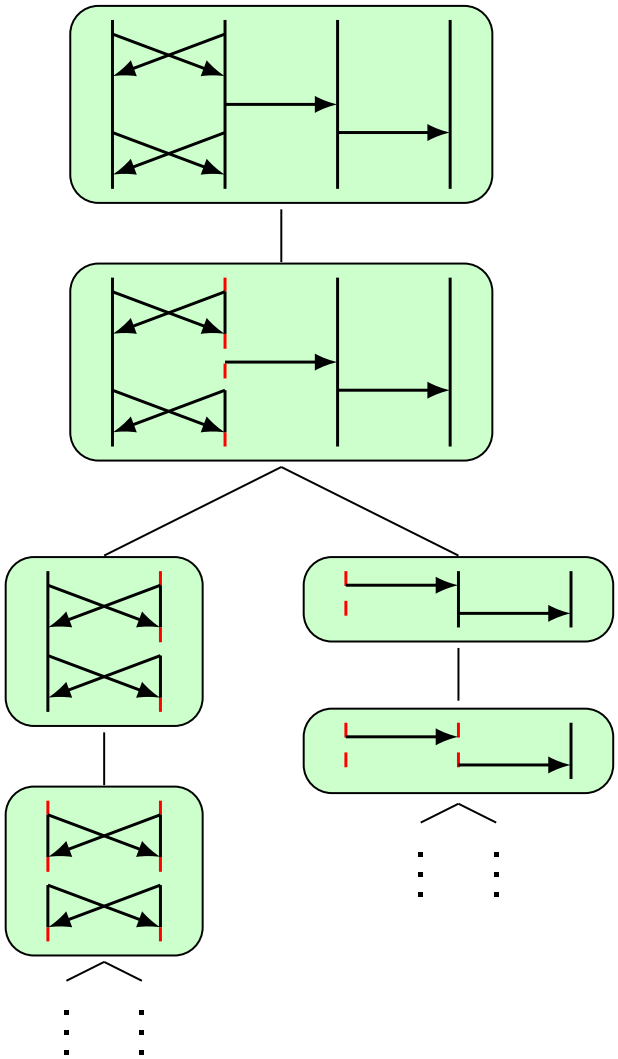
A k-context bounded MSCs over a pipeline topology has split-width at most $2k + 2$.



- At most k blocks on each process
- Messages detached with two extra cuts

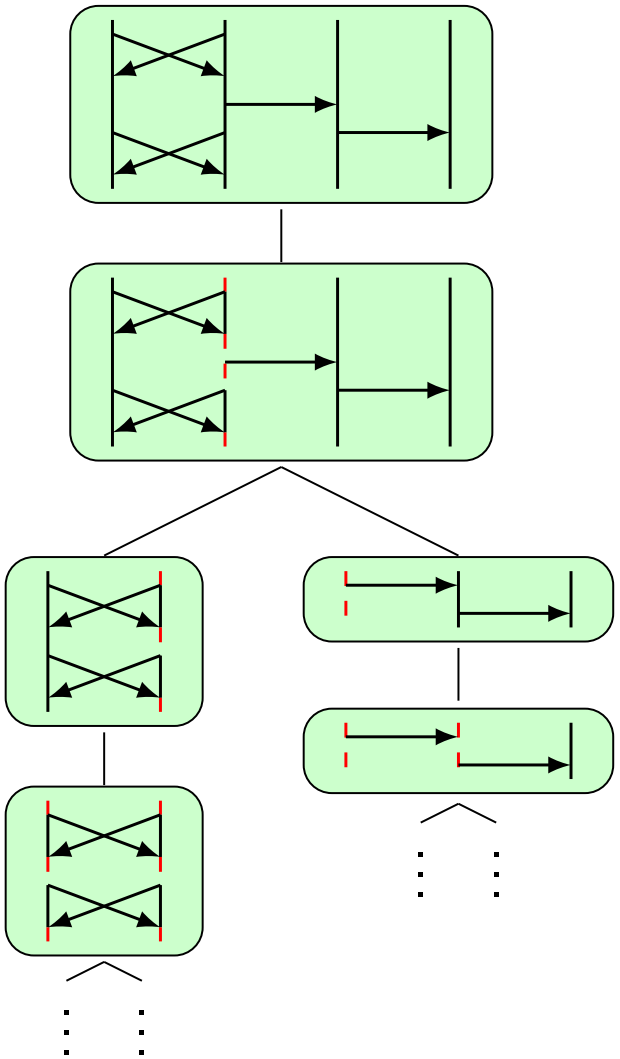
Interpretation into binary trees : k-DSTs

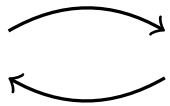
Eve's strategy = a tree



Interpretation into binary trees : k-DSTs

Eve's strategy $=$ a tree





k-DST (Disambiguated Split-Term)

- Tree of identical structure
- Finite alphabet, depending only on k

Interpretation into binary trees : k-DSTs

- k-DST t $\longmapsto$ unique MSC M_t of split-width $\leq k$

Interpretation into binary trees : k-DSTs

- k-DST t $\longmapsto$ unique MSC M_t of split-width $\leq k$
- MSC of split-width $\leq k$ $\longmapsto$ set of k-DSTs
  (over any topology)

Decision procedure

$\mathcal{C}$-Model-Checking (parameterized)

INPUT: $\mathcal{S}$, φ
QUESTION: $L(\mathcal{S}) \cap \mathcal{C} \subseteq L(\varphi)$?

Decision procedure

$\mathcal{C}$-Model-Checking (parameterized)

INPUT: $\mathcal{S}$, φ
QUESTION: $L(\mathcal{S}) \cap \mathcal{C} \subseteq L(\varphi)$?

- Assume $\mathcal{C}$ has split-width bounded by k, and there is a tree-automaton $\mathcal{A}^k_{\mathcal{C}}$ such that $\mathcal{C} = \{M_t \mid t \in L(\mathcal{A}^k_{\mathcal{C}})\}$.

Decision procedure

$\mathcal{C}$-Model-Checking (parameterized)

INPUT: $\mathcal{S}$, φ
QUESTION: $L(\mathcal{S}) \cap \mathcal{C} \subseteq L(\varphi)$?

- Assume $\mathcal{C}$ has split-width bounded by k, and there is a tree-automaton $\mathcal{A}^k_{\mathcal{C}}$ such that $\mathcal{C} = \{M_t \mid t \in L(\mathcal{A}^k_{\mathcal{C}})\}$.
- From $\mathcal{S}$, compute a tree-automaton $\mathcal{A}^k_{\mathcal{S}}$ such that $L(\mathcal{A}^k_{\mathcal{S}}) = \{t \mid M_t \in L(\mathcal{S})\}$. $\mathcal{A}^k_{\mathcal{S}}$ has size $|\mathcal{S}|^{\text{poly(k)}}$.

Decision procedure

$\mathcal{C}$-Model-Checking (parameterized)

INPUT: $\mathcal{S}$, φ
QUESTION: $L(\mathcal{S}) \cap \mathcal{C} \subseteq L(\varphi)$?

- Assume $\mathcal{C}$ has split-width bounded by k, and there is a tree-automaton $\mathcal{A}^k_{\mathcal{C}}$ such that $\mathcal{C} = \{M_t \mid t \in L(\mathcal{A}^k_{\mathcal{C}})\}$.
- From $\mathcal{S}$, compute a tree-automaton $\mathcal{A}^k_{\mathcal{S}}$ such that $L(\mathcal{A}^k_{\mathcal{S}}) = \{t \mid M_t \in L(\mathcal{S})\}$. $\mathcal{A}^k_{\mathcal{S}}$ has size $|\mathcal{S}|^{\text{poly(k)}}$.
- From φ, compute a tree-automaton $\mathcal{A}^k_{\varphi}$ such that $L(\mathcal{A}^k_{\varphi}) = \{t \mid M_t \in L(\varphi)\}$. $\mathcal{A}^k_{\varphi}$ is of exponential (PDL) to non-elementary (MSO) size, and only polynomial in $|\mathcal{S}|$.

Decision procedure

$\mathcal{C}$-Model-Checking (parameterized)

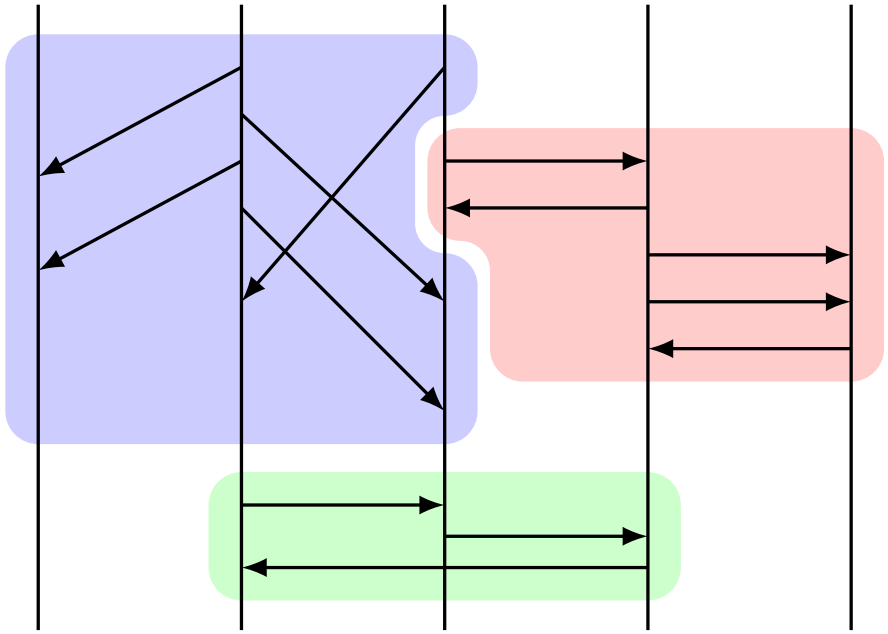
INPUT: $\mathcal{S}$, φ
QUESTION: $L(\mathcal{S}) \cap \mathcal{C} \subseteq L(\varphi)$? $\quad \rightsquigarrow L(\mathcal{A}^k_{\mathcal{S}} \cap \mathcal{A}^k_{\mathcal{C}} \cap \mathcal{A}^k_{\neg\varphi}) = \varnothing$?

- Assume $\mathcal{C}$ has split-width bounded by k, and there is a tree-automaton $\mathcal{A}^k_{\mathcal{C}}$ such that $\mathcal{C} = \{M_t \mid t \in L(\mathcal{A}^k_{\mathcal{C}})\}$.
- From $\mathcal{S}$, compute a tree-automaton $\mathcal{A}^k_{\mathcal{S}}$ such that $L(\mathcal{A}^k_{\mathcal{S}}) = \{t \mid M_t \in L(\mathcal{S})\}$. $\mathcal{A}^k_{\mathcal{S}}$ has size $|\mathcal{S}|^{\text{poly(k)}}$.
- From φ, compute a tree-automaton $\mathcal{A}^k_{\varphi}$ such that $L(\mathcal{A}^k_{\varphi}) = \{t \mid M_t \in L(\varphi)\}$. $\mathcal{A}^k_{\varphi}$ is of exponential (PDL) to non-elementary (MSO) size, and only polynomial in $|\mathcal{S}|$.

Concrete classes of bounded split-width

- Context-bounded MSCs over topologies of bounded tree-width
- Tile-bounded MSCs over topologies of bounded tree-width
- "Existentially-bounded" MSCs

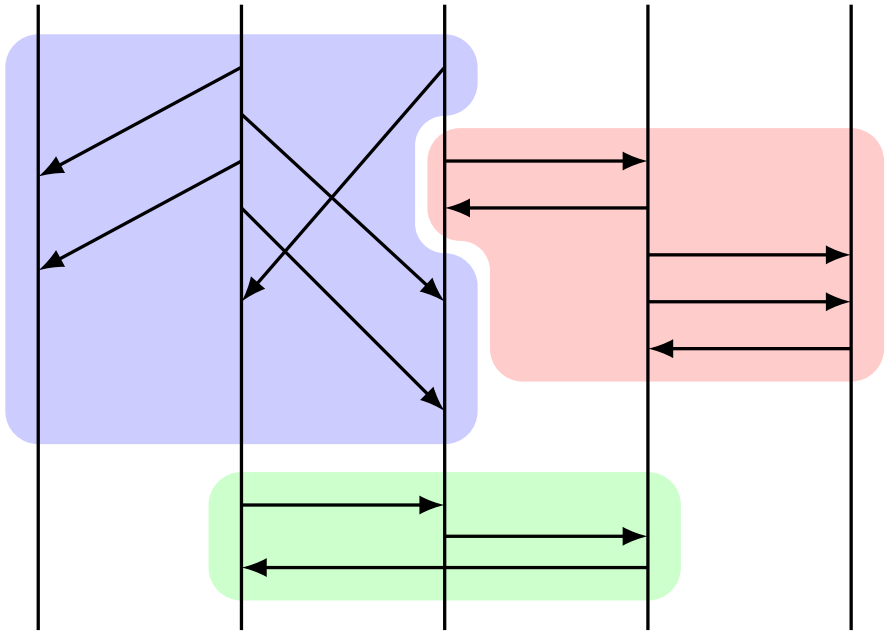
Tile-bounded MSCs



A decomposition for $k = 5, \ell = 3$

- No message edge between two distinct tiles
- A tile has at most k processes
- A tile has split-width at most k
- At most ℓ tiles on each process

Tile-bounded MSCs



A decomposition for $k = 5, \ell = 3$

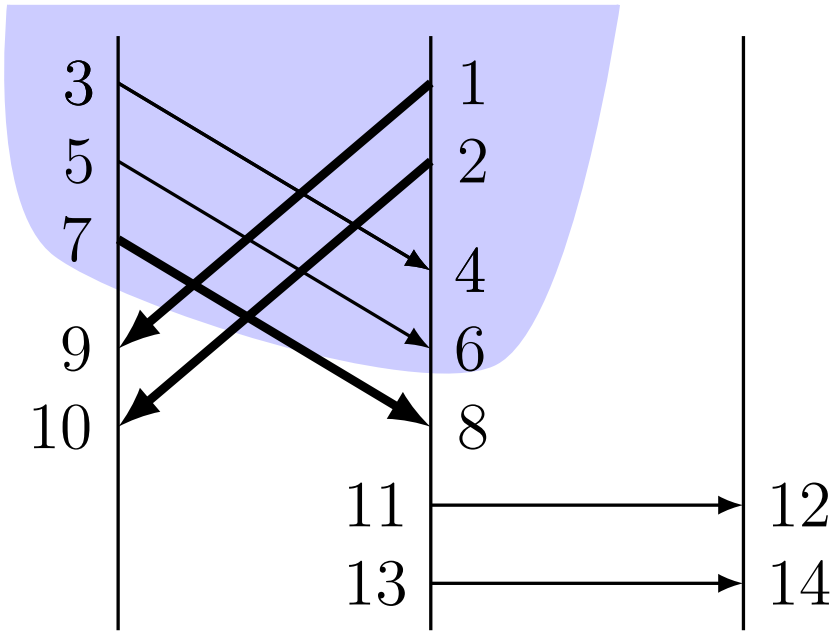
- No message edge between two distinct tiles
- A tile has at most k processes
- A tile has split-width at most k
- At most ℓ tiles on each process

(tile-bounded + topology of bounded tree-width)
$\iff$ (bounded split-width)

Existentially bounded MSCs

Fixed topology

At any time, $\leq k$ pending messages
(for some linearization).

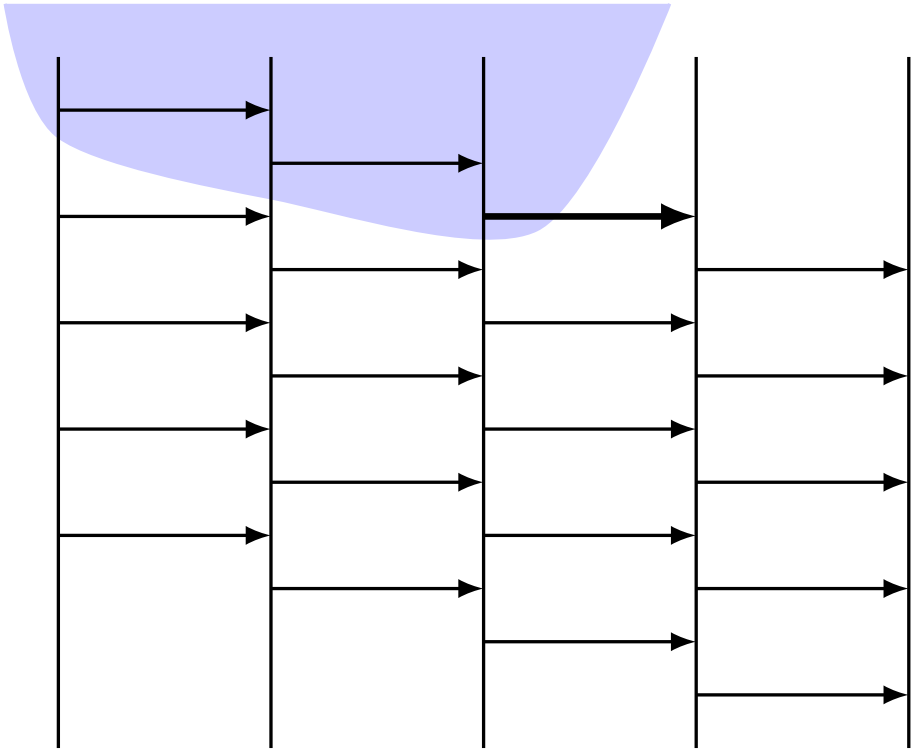


existentially 3-bounded

Existentially bounded MSCs

Parameterized case

A any time, $\leq k$ pending messages (for some linearization).

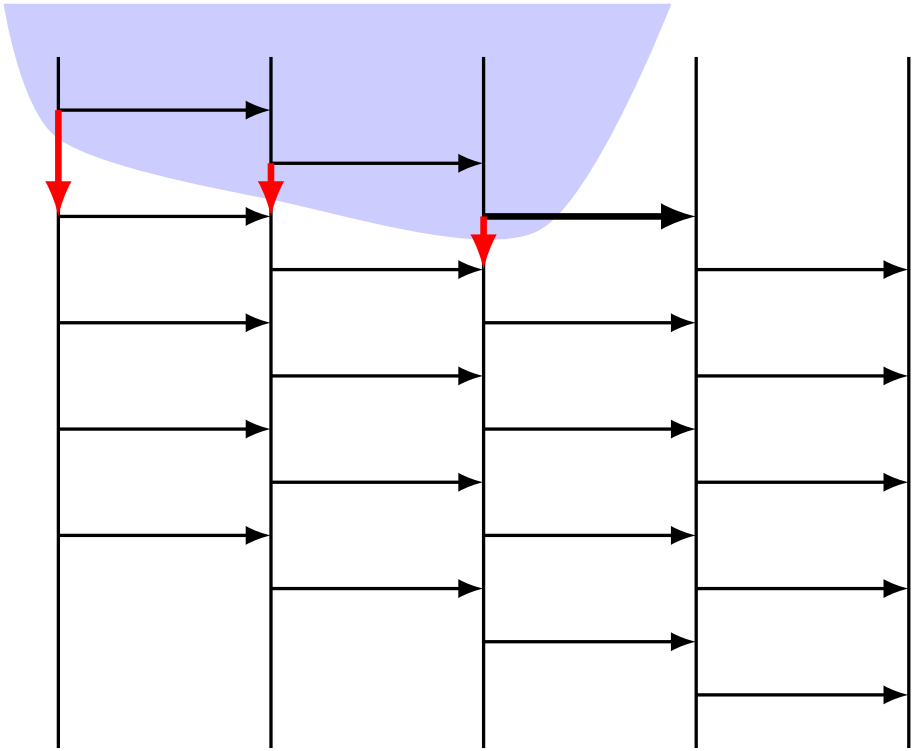


- existentially 1-bounded
- Undecidable

Existentially bounded MSCs

Parameterized case

A any time, $\leq k$ pending messages or process edges (for some linearization).



- ~~existentially 1-bounded~~
- Undecidable

Decision procedure

$\mathcal{C}$-Model-Checking (parameterized)

INPUT: $\mathcal{S}$, φ
QUESTION: $L(\mathcal{S}) \cap \mathcal{C} \subseteq L(\varphi)$? $\quad \rightsquigarrow L(\mathcal{A}^k_{\mathcal{S}} \cap \mathcal{A}^k_{\mathcal{C}} \cap \mathcal{A}^k_{\neg\varphi}) = \varnothing$?

- Assume $\mathcal{C}$ has split-width bounded by k, and there is a tree-automaton $\mathcal{A}^k_{\mathcal{C}}$ such that $\mathcal{C} = \{M_t \mid t \in L(\mathcal{A}^k_{\mathcal{C}})\}$.
- From $\mathcal{S}$, compute a tree-automaton $\mathcal{A}^k_{\mathcal{S}}$ such that $L(\mathcal{A}^k_{\mathcal{S}}) = \{t \mid M_t \in L(\mathcal{S})\}$. $\mathcal{A}^k_{\mathcal{S}}$ has size $|\mathcal{S}|^{\text{poly(k)}}$.
- From φ, compute a tree-automaton $\mathcal{A}^k_{\varphi}$ such that $L(\mathcal{A}^k_{\varphi}) = \{t \mid M_t \in L(\varphi)\}$. $\mathcal{A}^k_{\varphi}$ is of exponential (PDL) to non-elementary (MSO) size, and only polynomial in $|\mathcal{S}|$.

Results

	$\mathcal{C}_k = \mathsf{EB}_k$	$\mathcal{C}_{(k,h)} = \mathsf{CB}_{k,h}$	$\mathcal{C}_k = \mathsf{SW}_k$	$\mathcal{C}_{(k,\ell,h)} = \mathsf{TB}_{k,\ell,h}$
$\mathcal{C}$-Reachability	Pspace-c	Exptime-c		
$\mathcal{C}$-Emptiness				
$\mathcal{C}$-CPDL-Sat/MC				
$\mathcal{C}$-ICPDL-Sat/MC	Expspace-c	2-Exptime-c		
$\mathcal{C}$-MSO-Sat/MC	Non elementary			

Polynomial in $|\mathcal{S}|$.

Conclusion

- Generalisation of split-width to the parameterized case.
- Sufficient conditions for bounded split-width.
- Optimal complexities for all classes. Polynomial in the size of the automaton.

Conclusion

- Generalisation of split-width to the parameterized case.
- Sufficient conditions for bounded split-width.
- Optimal complexities for all classes. Polynomial in the size of the automaton.

- Other classes of bounded split-width?
- Data, process identifiers?
- Dynamic creation of processes?