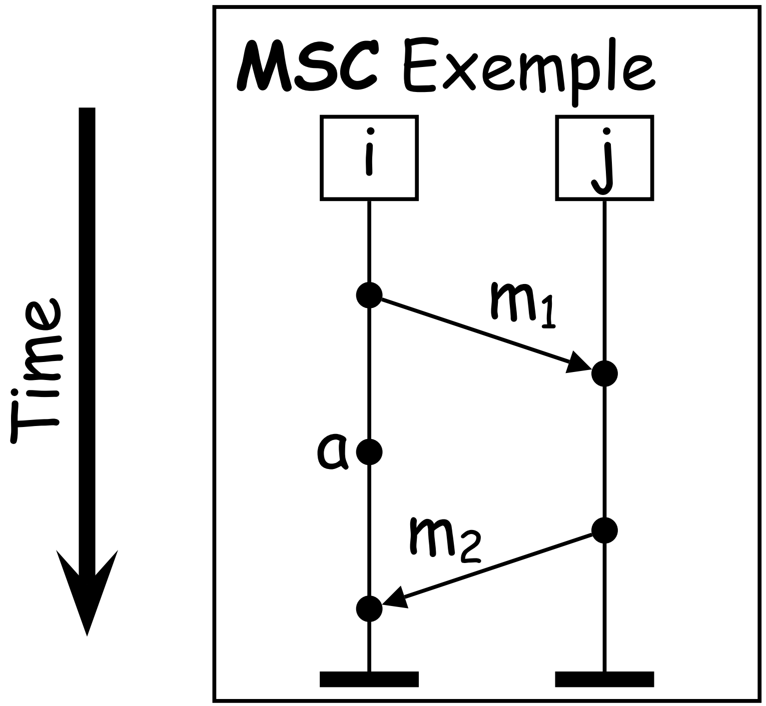
Checking MSC graphs with Petri nets

Rémi MORIN

Laboratoire d'Informatique Fondamentale de Marseille

Basic MSCs (ITU Z120)



msc Exemple;

instance i;

out m_1 **to** j;

action a;

in m_2 **from** j;

endinstance;

instance j;

in m_1 **from** i;

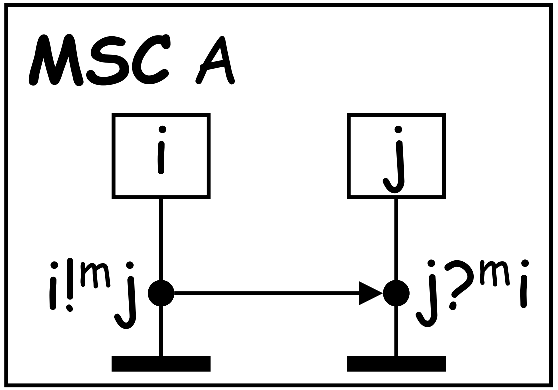
out m_2 **to** i;

endinstance;

endmsc;

Each send corresponds to a receive, and vice versa.

MSCs are regarded as labeled partial orders



In this talk, we do not distinguish message types.

As usual, we assume **FIFO communication**.

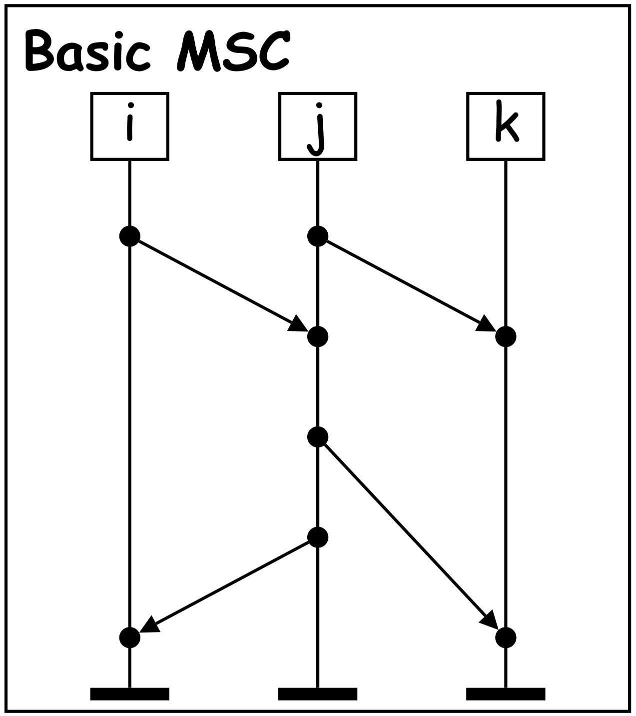
Compositional MSCs [Gunter et al, TACAS 2001]

Consider a basic MSC $M = (E, \preccurlyeq, \xi)$.

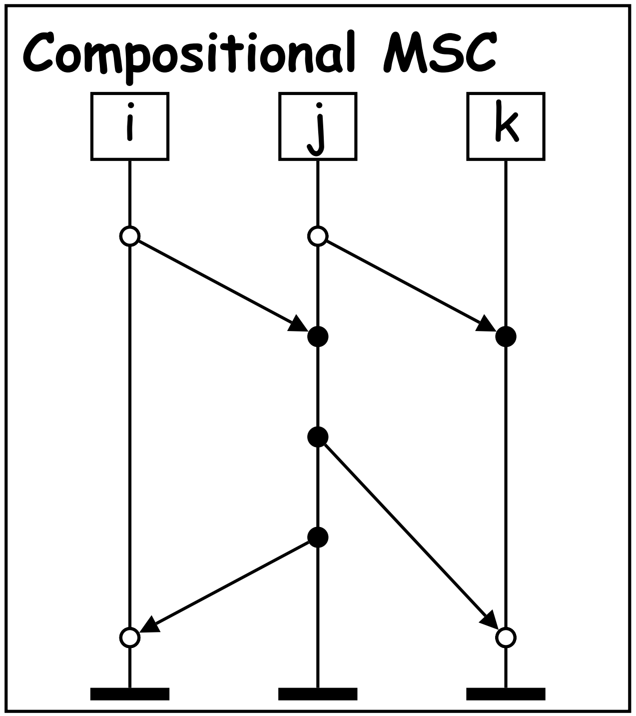
Let Past $\subseteq E$ be a **prefix** consisting of send events.

Let Future $\subseteq E$ be a **suffix** consisting of receive events.

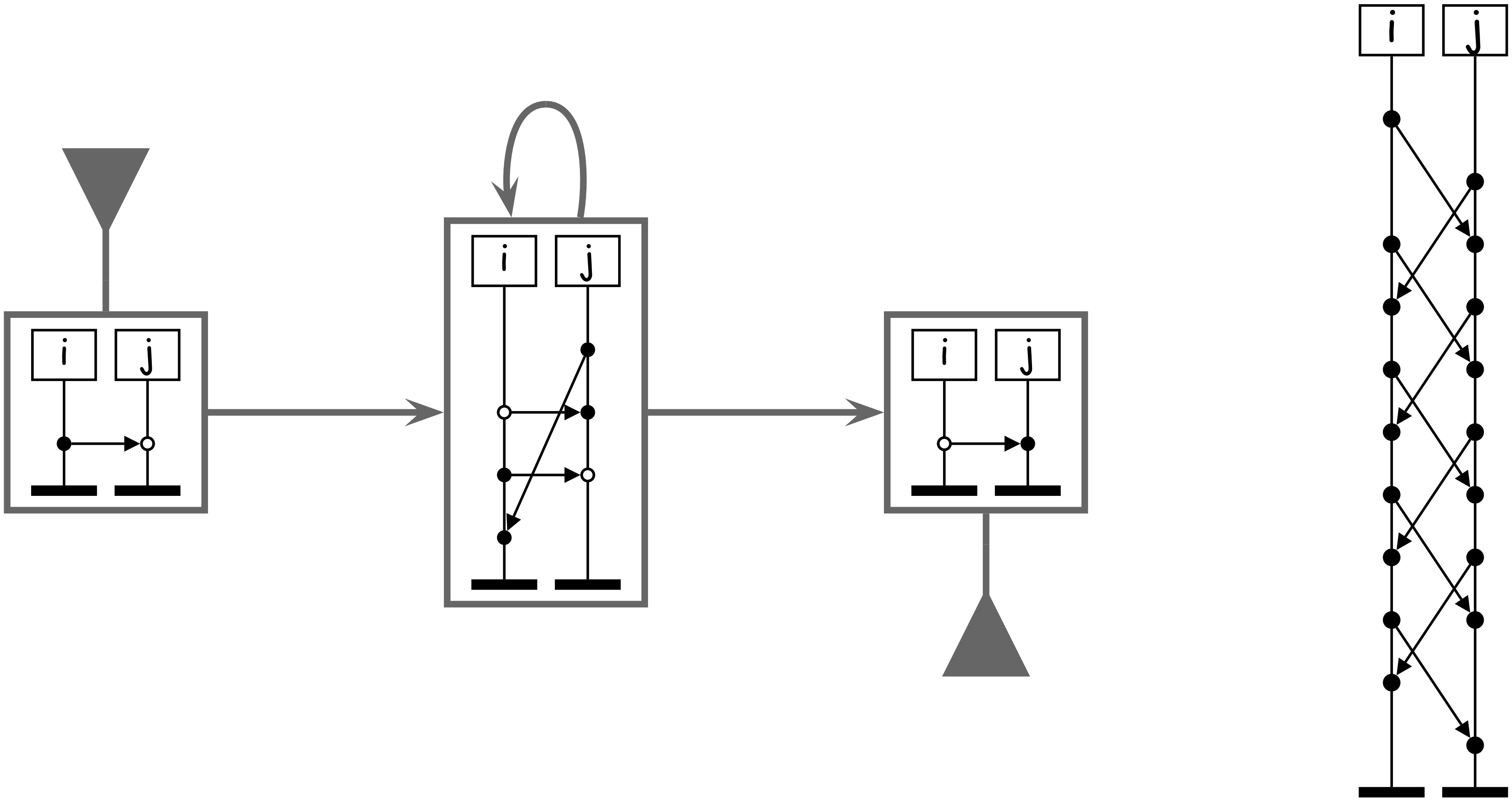
Then $(E \setminus (\text{Past} \cup \text{Future}), \preccurlyeq, \xi)$ is a (compositional) MSC.



$\rightsquigarrow$

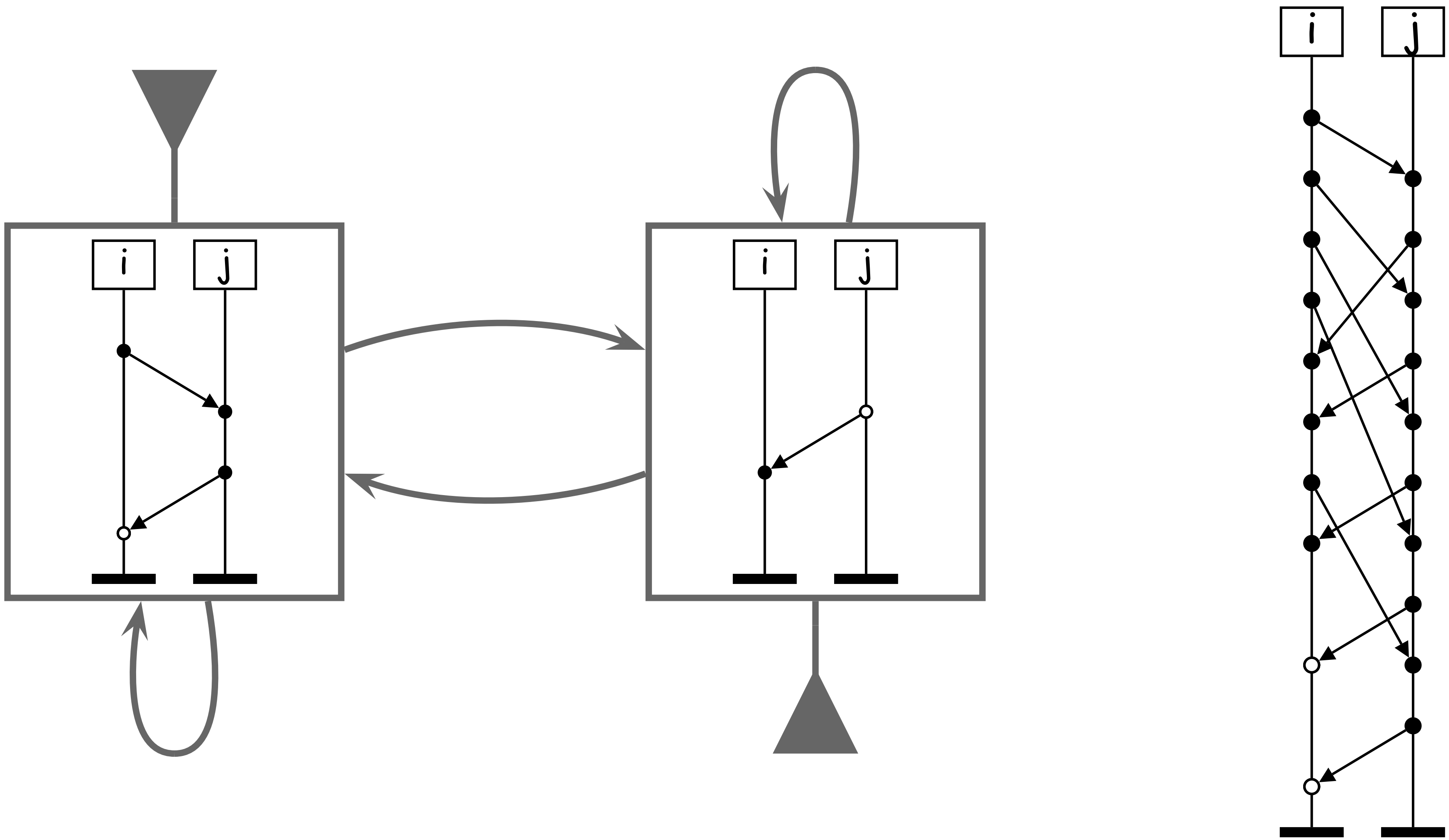


The alternating bit protocol as an MSG

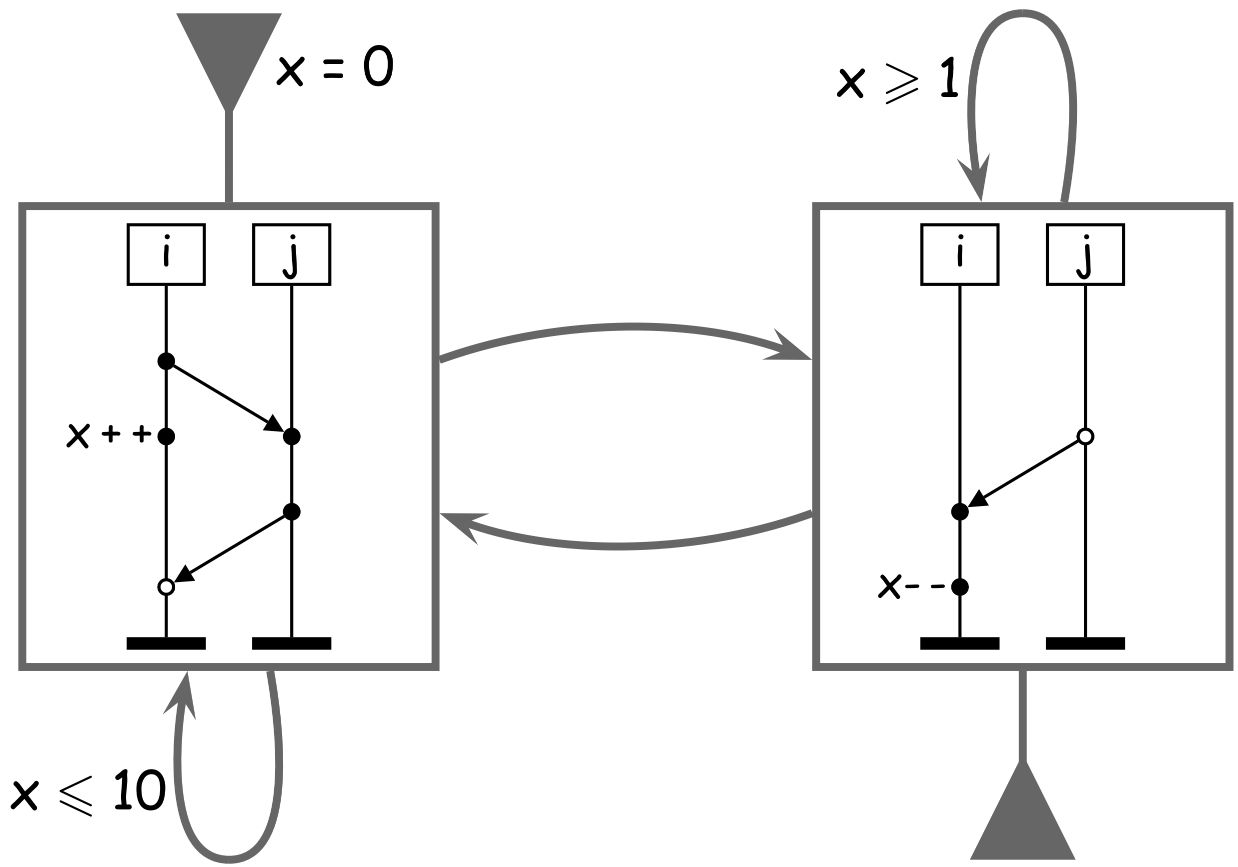


Each accepted path of this MSG corresponds to a basic MSC.

The sliding window protocol as a non-safe MSG

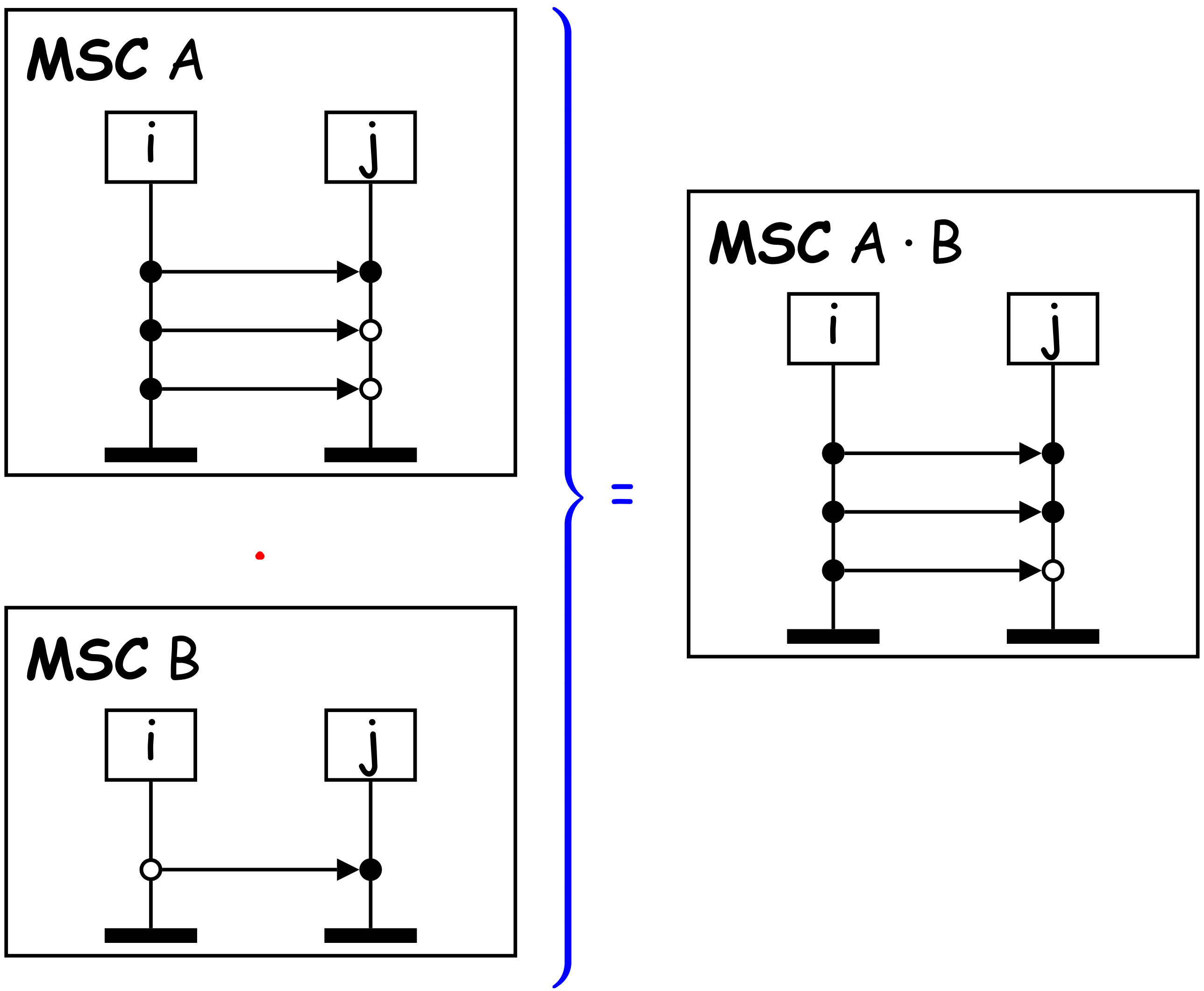


Protocol with a bounded counter



- ✔ Background
- ☛ Composition of compositional MSCs
- Emptiness is undecidable
- MSGs without matching arrow
- The subclass of linear MSGs
- Conclusion

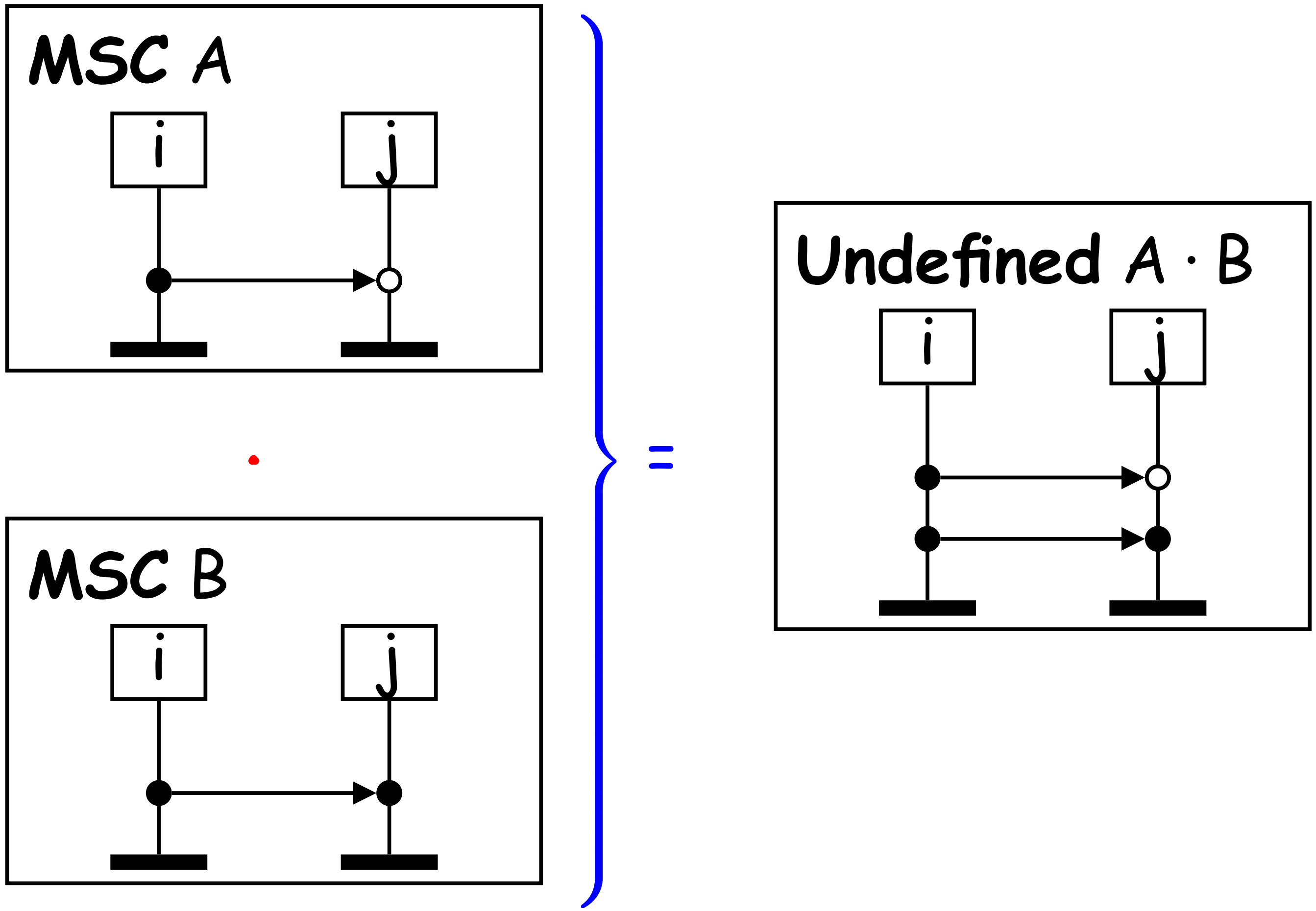
Product of two compositional MSCs



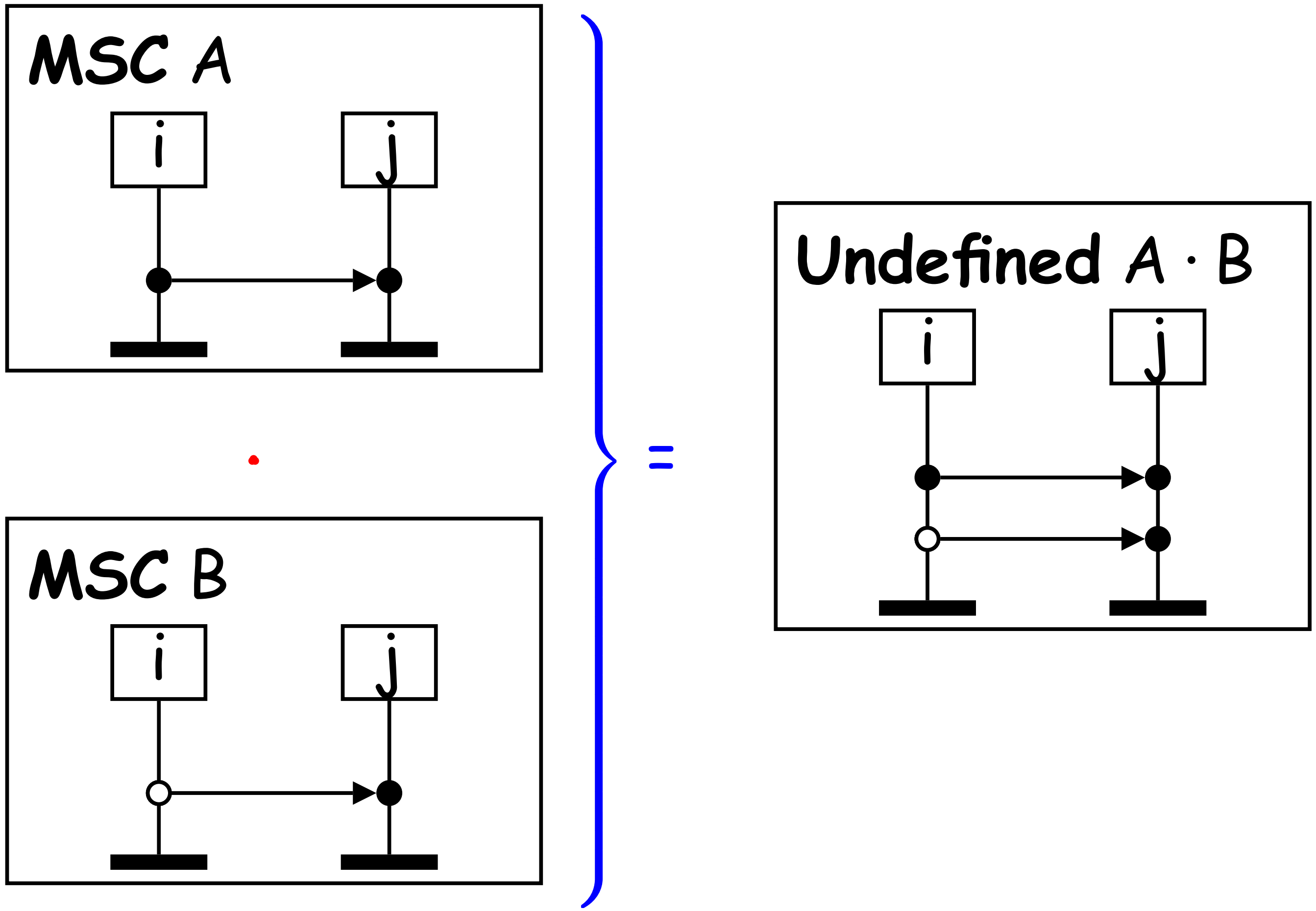
The events of A precede the events of B for each process.

The n-th send in A **matches** the n-th receive for each channel.

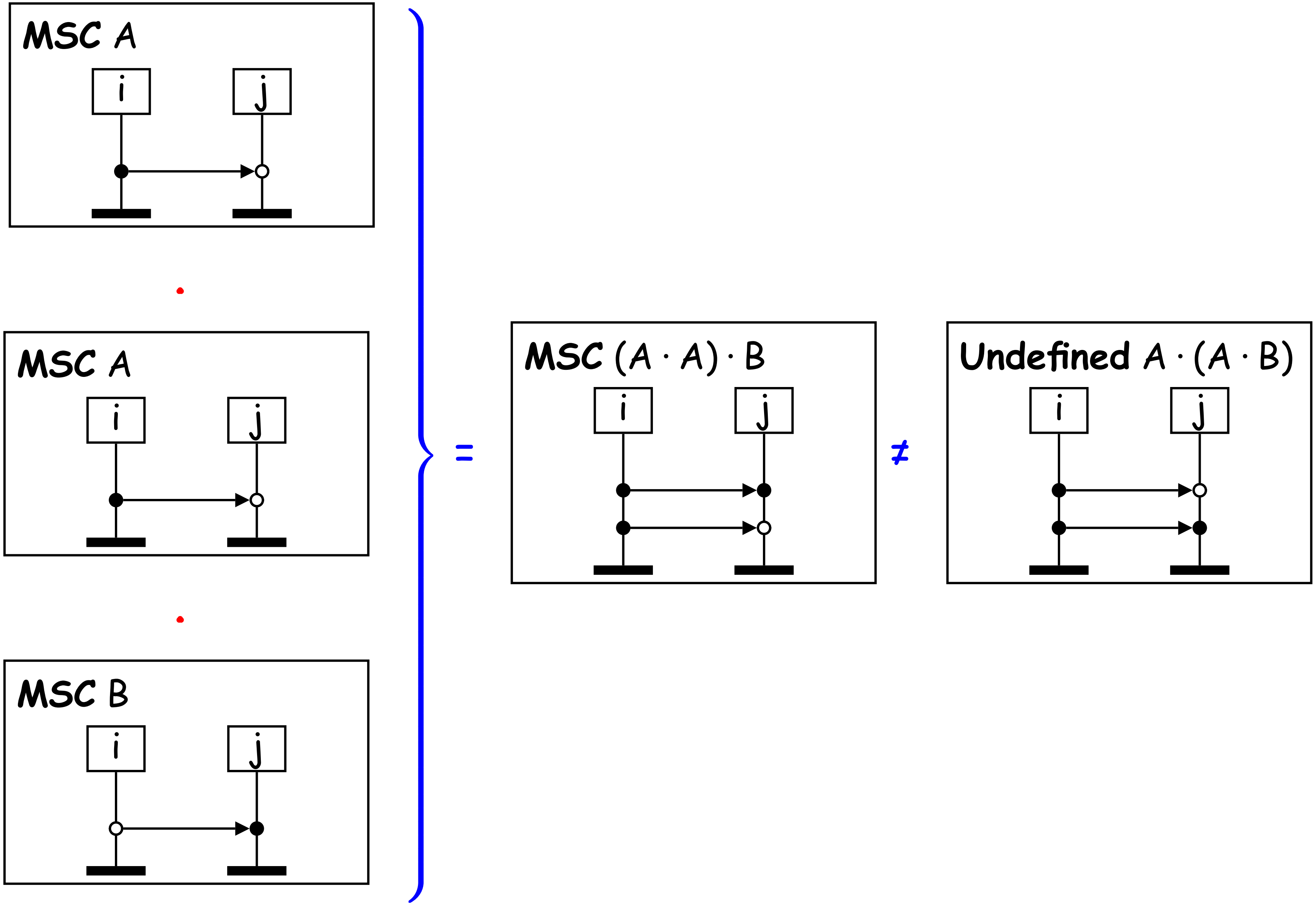
Some products are obviously undefined (1/2)



Some products are obviously undefined (2/2)

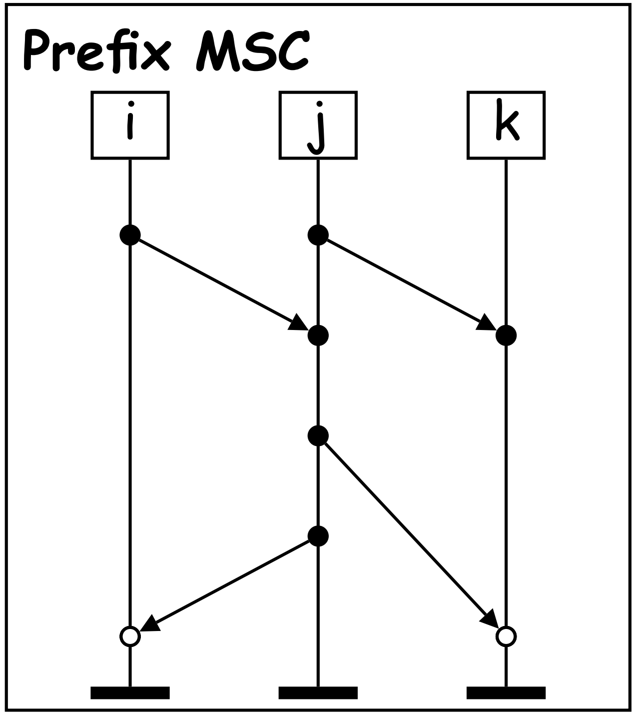


This product is not associative



Prefix MSCs (usually called "left-closed")

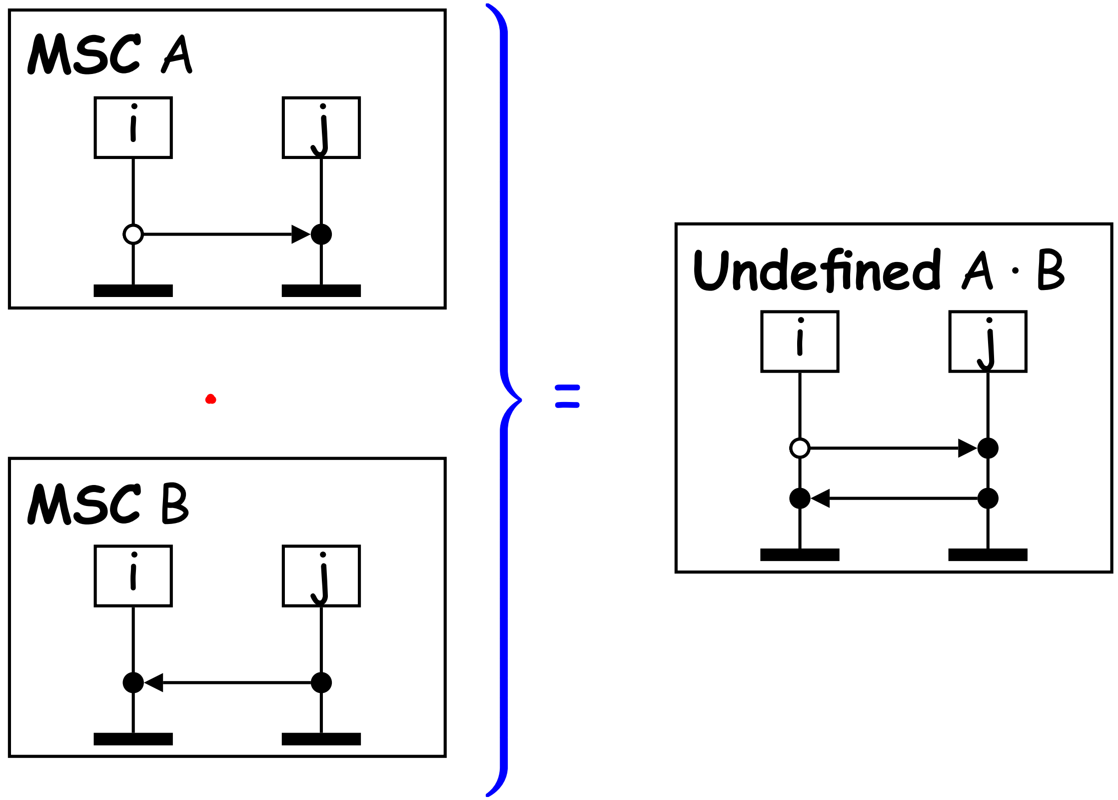
A prefix MSC is an MSC **without unmatched receive**.



It corresponds intuitively to an execution starting from the configuration where all channels are empty: **There is no pending message initially**.

Some restrictions (1/2)

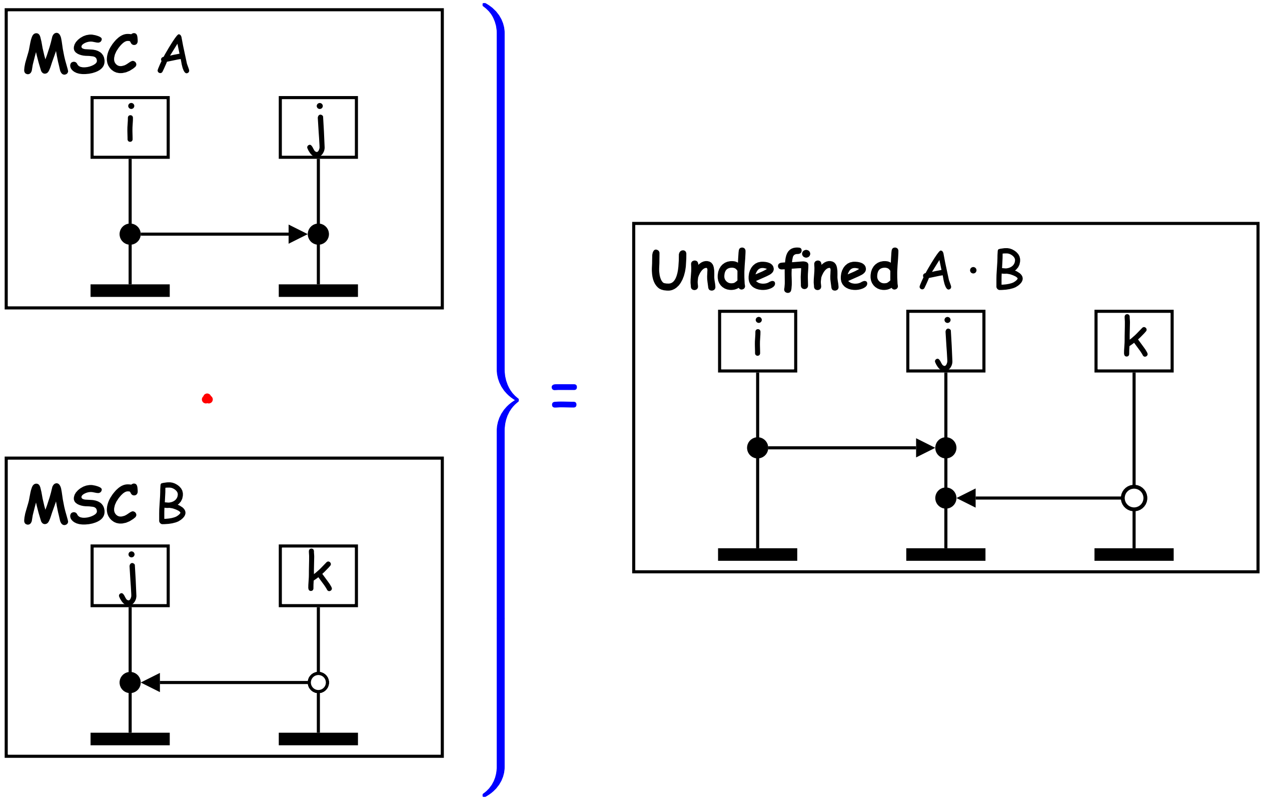
A must be a **prefix MSC**; otherwise $A \cdot B$ is undefined.



This restriction originates with [Gunter et al, TACAS 2001].

Some restrictions (2/2)

$A \cdot B$ must be a **prefix MSC**; otherwise $A \cdot B$ is undefined.

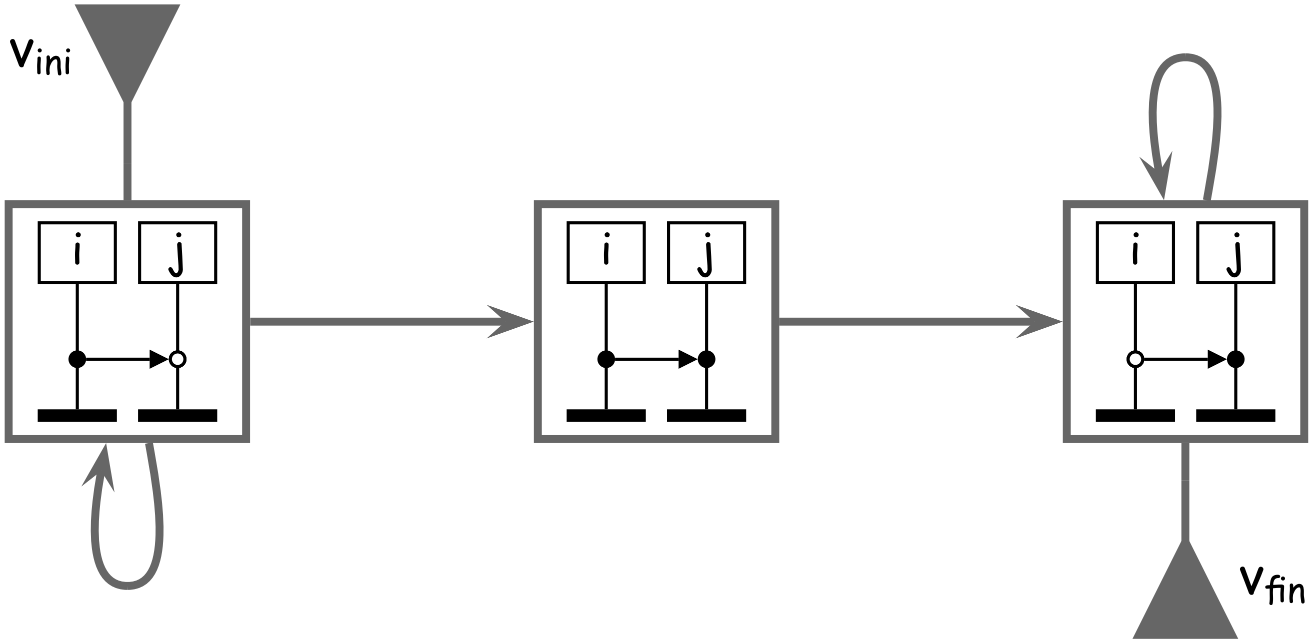


This restriction originates with [Gunter et al, TACAS 2001], too.

Language $\mathcal{L}(G)$ of accepted MSCs

A path $v_0, \ldots, v_n$ from the initial vertex $v_0 = v_{ini}$ is valid if the product $msc(v_0) \cdot \ldots \cdot msc(v_n)$ is defined.

The language $\mathcal{L}(G)$ collects the product MSCs of all valid paths that reach the final vertex v_{fin}.



What is $\mathcal{L}(G)$ in this case?

- ✔ Background
- ✔ Composition of compositional MSCs
- ☛ Emptiness is undecidable
- MSGs without matching arrow
- The subclass of linear MSGs
- Conclusion

2-counter machine [Minsky, 1967]

A 2-counter machine is an abstract machine made of

- two unbounded counters c_1 and c_2 that can hold a non-negative integer (the memory)
- a sequence of labeled instructions (the program).

The allowed instructions are

- "c_1**++**" or "c_2**++**" increments the value of the counter;
- "**if** c_1 **= 0 goto** l' **else** c_1**--**" transfers control to the instruction labeled by l' if c_1 equals zero, and otherwise decrements c_1 and continues with the next instruction.
- "**if** c_2 **= 0 goto** l' **else** c_2**--**"

2-counter machine [Minsky, 1967]

The initial value of both counters is 0.

A 2-counter machine is deterministic: It will

- either reach the last instruction of its program and halt after a **finite** number of steps
- or perform an **infinite computation**.

It is undecidable whether a given 2-counter machine halts.

> **Theorem**
>
> The emptiness problem $\mathcal{L}(G) = \emptyset$ is undecidable.

From a 2-counter machine to an MSG (1/3)

We build an MSG G over three processes: i, c_1 and c_2 such that the three next properties are equivalent:

(i) $\mathcal{L}(G) \neq \emptyset$;

(ii) Some **valid path** reaches the final node;

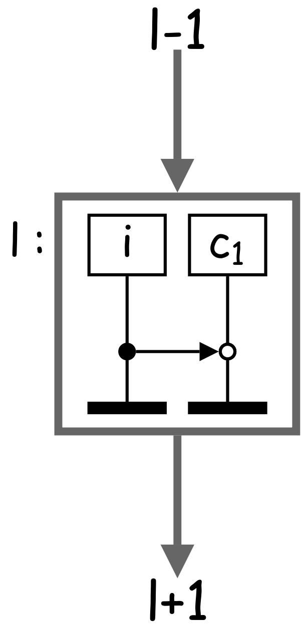
(iii) The 2-counter machine halts.

Each valid path of G corresponds to a computation of the machine, and vice versa.

The value of c_1 (resp. c_2) is encoded by the number of pending messages in the channel from i to c_1 (resp. c_2).

From a 2-counter machine to an MSG (2/3)

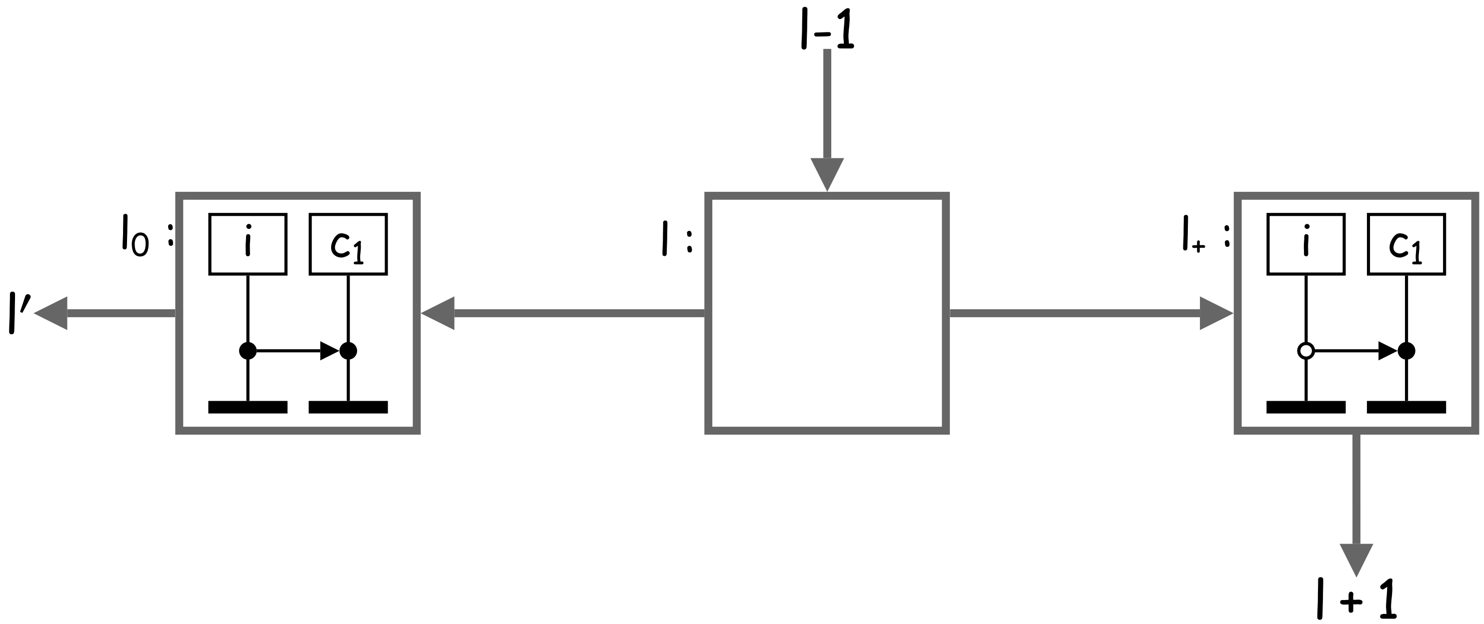
The labeled instruction "$l : c_1{+}{+}$" is encoded by a node l labeled by the MSC



For each prefix MSC M, the product $M \cdot msc(l)$ is defined.

From a 2-counter machine to an MSG (3/3)

The labeled instruction "l:**if** c_1 **= 0 goto** l' **else** c_1**--**"
is encoded by three nodes l, l_0 and l_+:



For each prefix MSC M, one and only one of the two products
$M \cdot msc(l_0)$ and $M \cdot msc(l_+)$ is defined.

Other undecidability results

1. An MSG is deadlock-free if each valid path can be completed into a valid accepted one.
   Deadlock-freeness is undecidable
   because G is deadlock-free iff the machine halts.

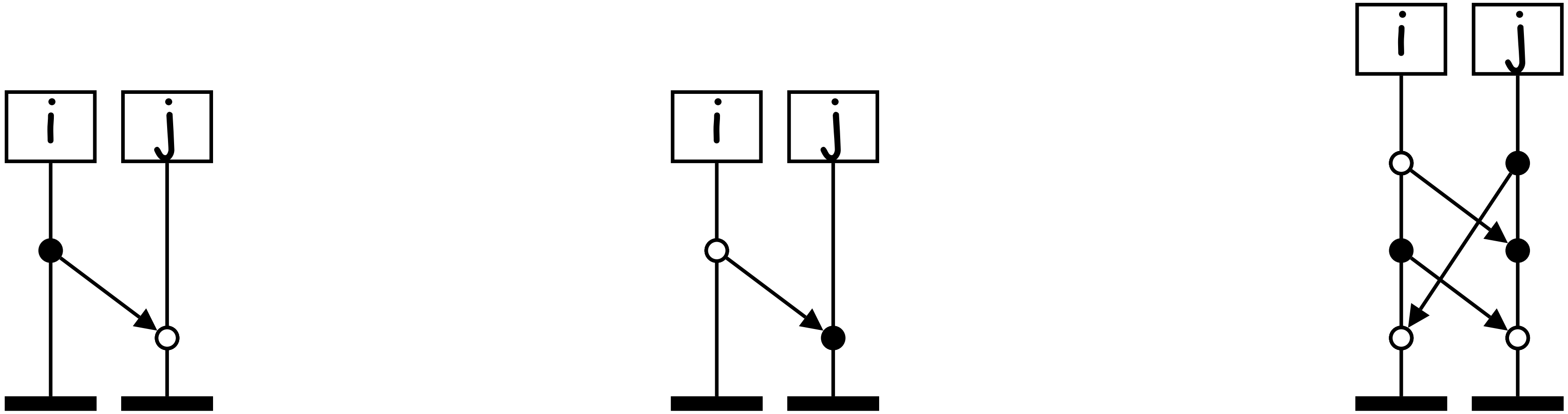
2. The use of a channel in a valid (resp. accepted) path is undecidable.

3. Boundedness is also undecidable.

- ✔ Background
- ✔ Composition of compositional MSCs
- ✔ Emptiness is undecidable
- ☛ MSGs without matching arrow
- The subclass of linear MSGs
- Conclusion

MSGs without matching arrow

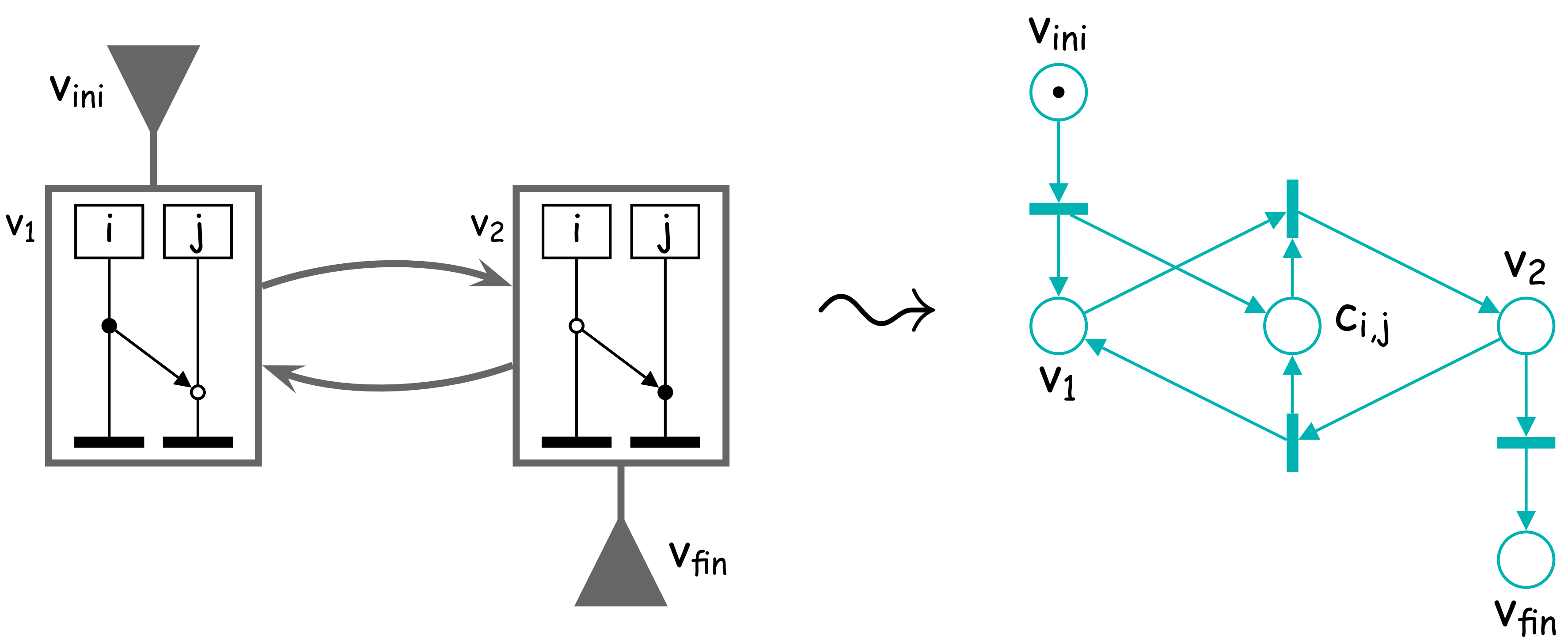
An MSG is without matching arrow if its nodes are labeled by MSCs with no complete message exchange, i.e. **all events are unmatched**.



Theorem

The emptiness problem for MSGs without matching arrow is equivalent to the covering problem for Petri nets.

From Emptiness to Covering (1/2)



The valid paths of G coincide with the firing sequences of N.

Some valid path reaches the final vertex v_{fin} if and only if some firing sequence puts a token in the place v_{fin}.

From Emptiness to Covering (2/2)

Let $G = (V, \longrightarrow, v_{ini}, v_{fin})$ be an MSG over the set of channels C.

Let $P = V \cup C$ be the set of places. We put $\mathfrak{m}_{ini} = \{v_{ini}\}$.

Each arc $v_1 \longrightarrow v_2$ in G corresponds to a rule $r_{v_1 \longrightarrow v_2}$ such that

(1) $^\bullet r_{v_1 \longrightarrow v_2}(v) = \begin{cases} 1 & \text{if } v = v_1 \\ 0 & \text{otherwise} \end{cases}$ and $r_{v_1 \longrightarrow v_2}{}^\bullet(v) = \begin{cases} 1 & \text{if } v = v_2 \\ 0 & \text{otherwise} \end{cases}$

(2) $^\bullet r_{v_1 \longrightarrow v_2}(c) = {}^\bullet msc(v_2)(c)$, i.e. the number of unmatched receives from c in $msc(v_2)$

(3) $r_{v_1 \longrightarrow v_2}{}^\bullet(c) = msc(v_2)^\bullet(c)$, i.e. the number of unmatched sends to c in $msc(v_2)$

From Covering to Emptiness (1/3)

Let N be a Petri net.

Let $\mathfrak{m}_{ini}, \mathfrak{m}_{fin}$ be two markings.

We can assume that:

- $\mathfrak{m}_{ini}$ contains a single token in the place p_{ini};
- $\mathfrak{m}_{fin}$ contains a single token in the place p_{fin}.

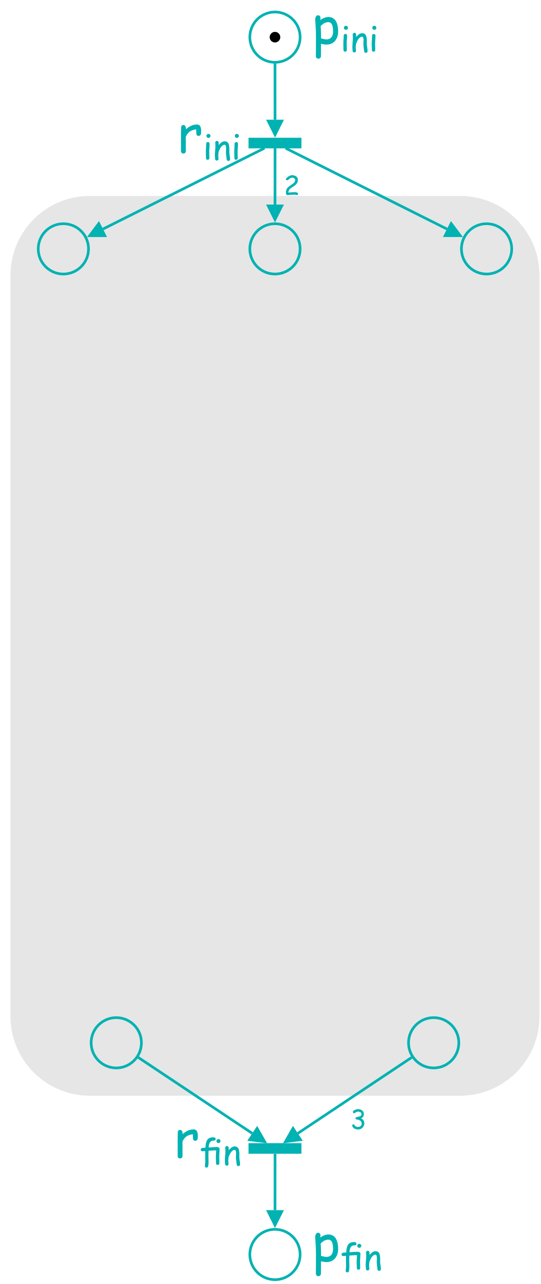
We build the MSG G with

- $\mathcal{I} = \{i\} \cup P$

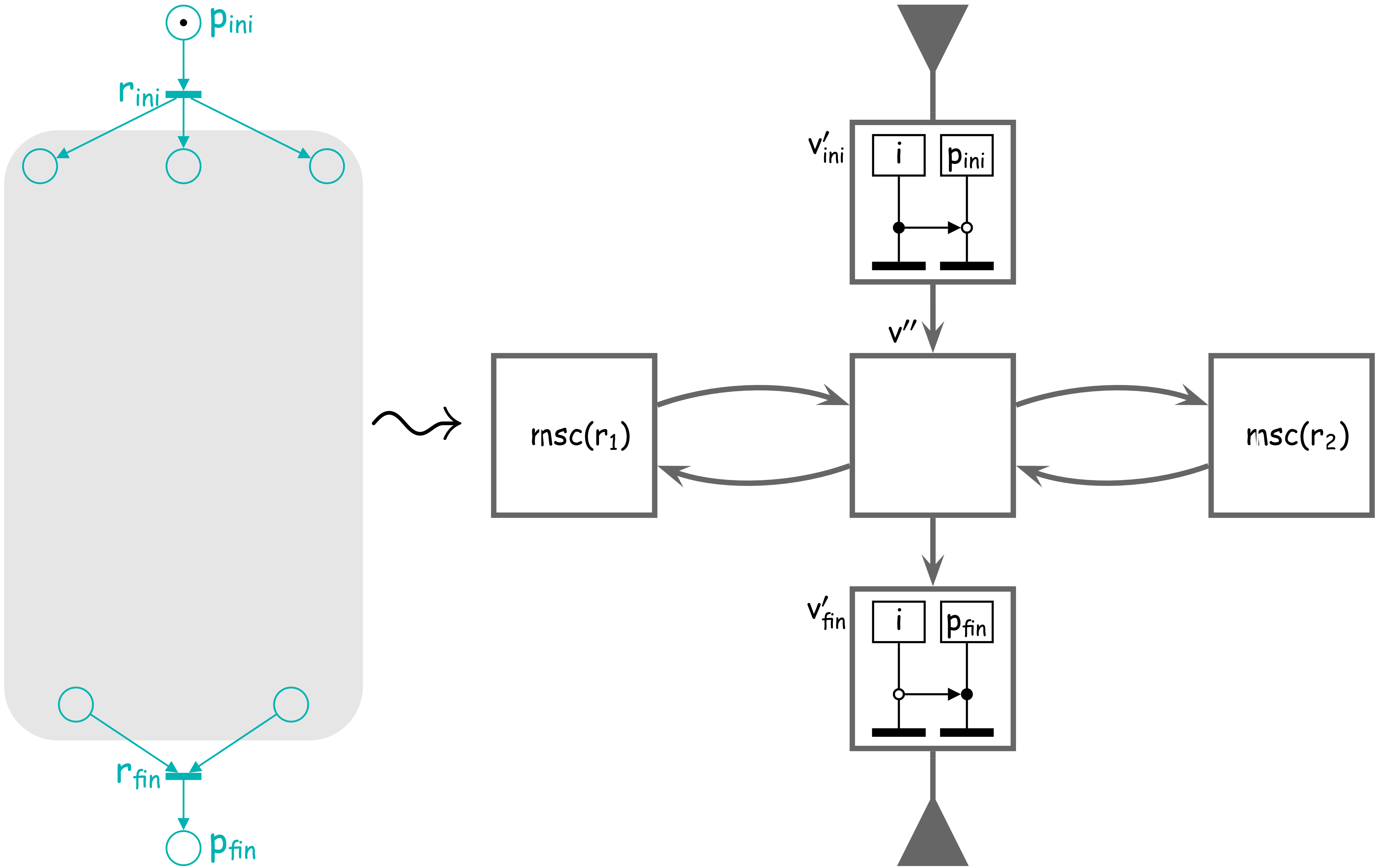
  Tokens in p are represented by messages from i to p.

- $V = \{v_{ini}, v_{fin}, v'_{ini}, v'_{fin}, v''\} \cup R$

  The 3 nodes v_{ini}, v_{fin} and v'' are labeled by the empty MSC.



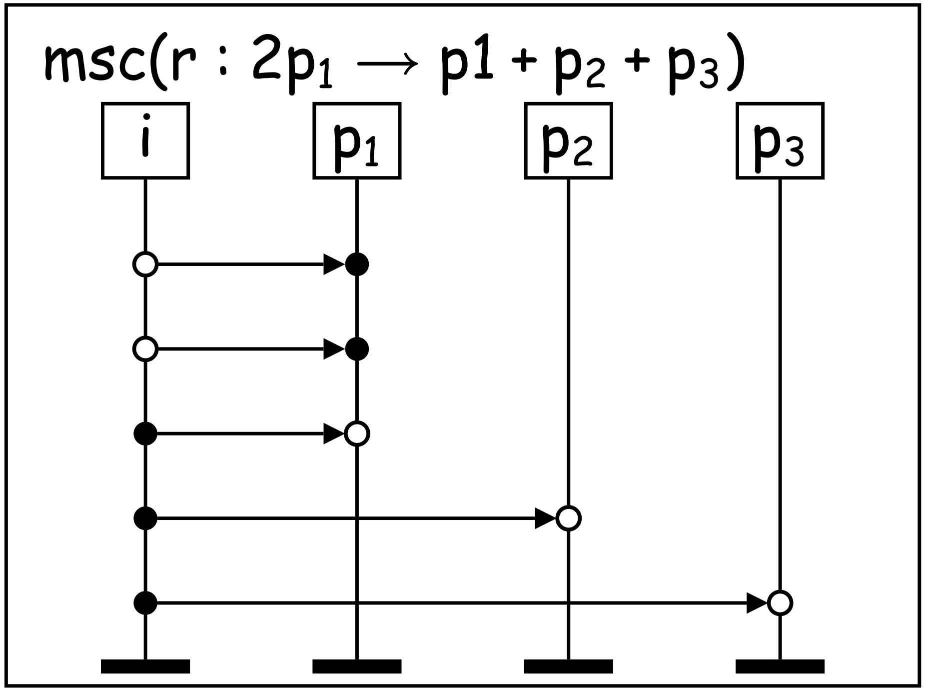
From Covering to Emptiness (2/3)



From Covering to Emptiness (3/3)

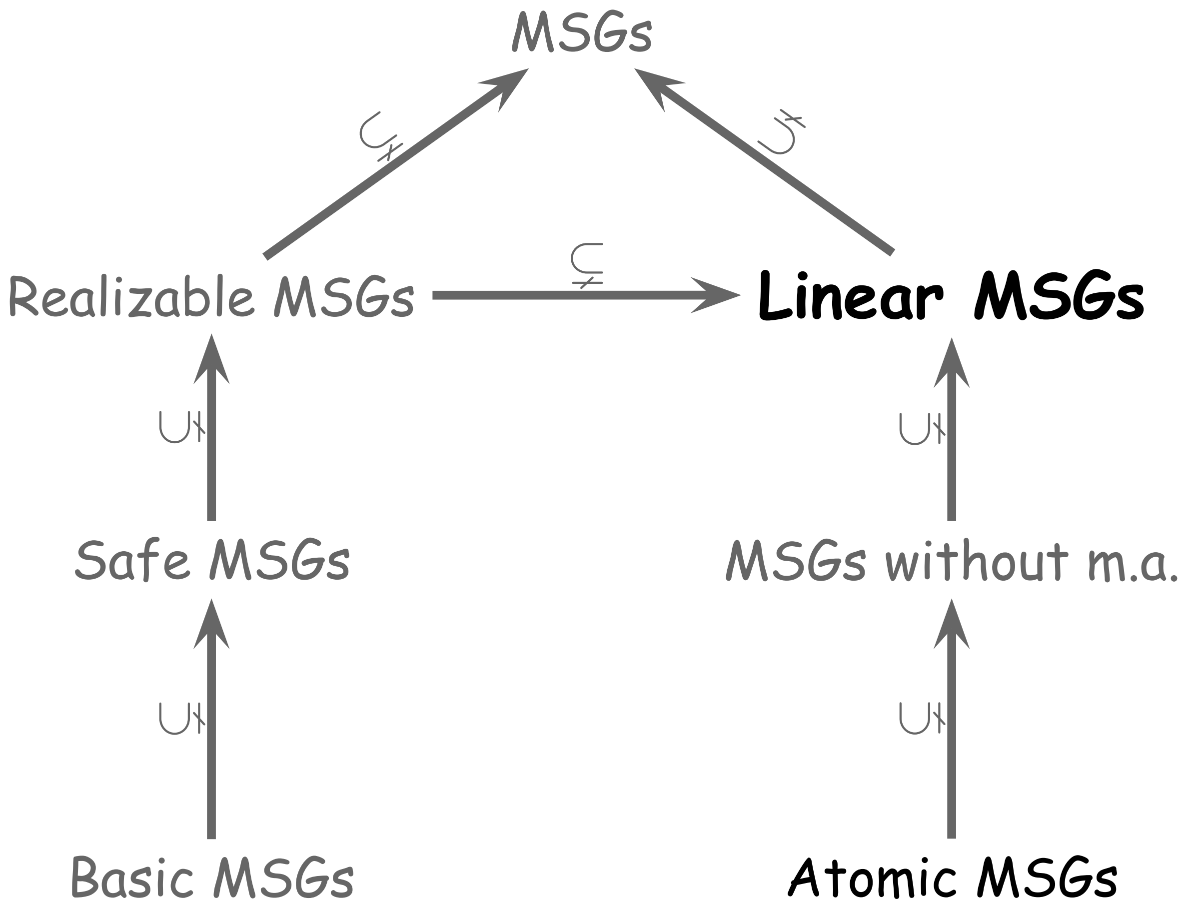
For each transition rule $r = ({}^\bullet r, r^\bullet)$, the node r is labeled by the MSC msc(r) and connected to v'' with two arcs:

- from v'' to v_r
- from v_r to v''



- ✔ Background
- ✔ Composition of compositional MSCs
- ✔ Emptiness is undecidable
- ✔ MSGs without matching arrow
- ☛ The subclass of linear MSGs
- Conclusion

Overview



MSGs without matching arrows and realizable MSGs are linear. Moreover the sliding window protocol is linear, too.

Linear MSGs

Definition

An MSG G is **linear** if its valid paths coincide with the firing sequences of the corresponding Petri net.

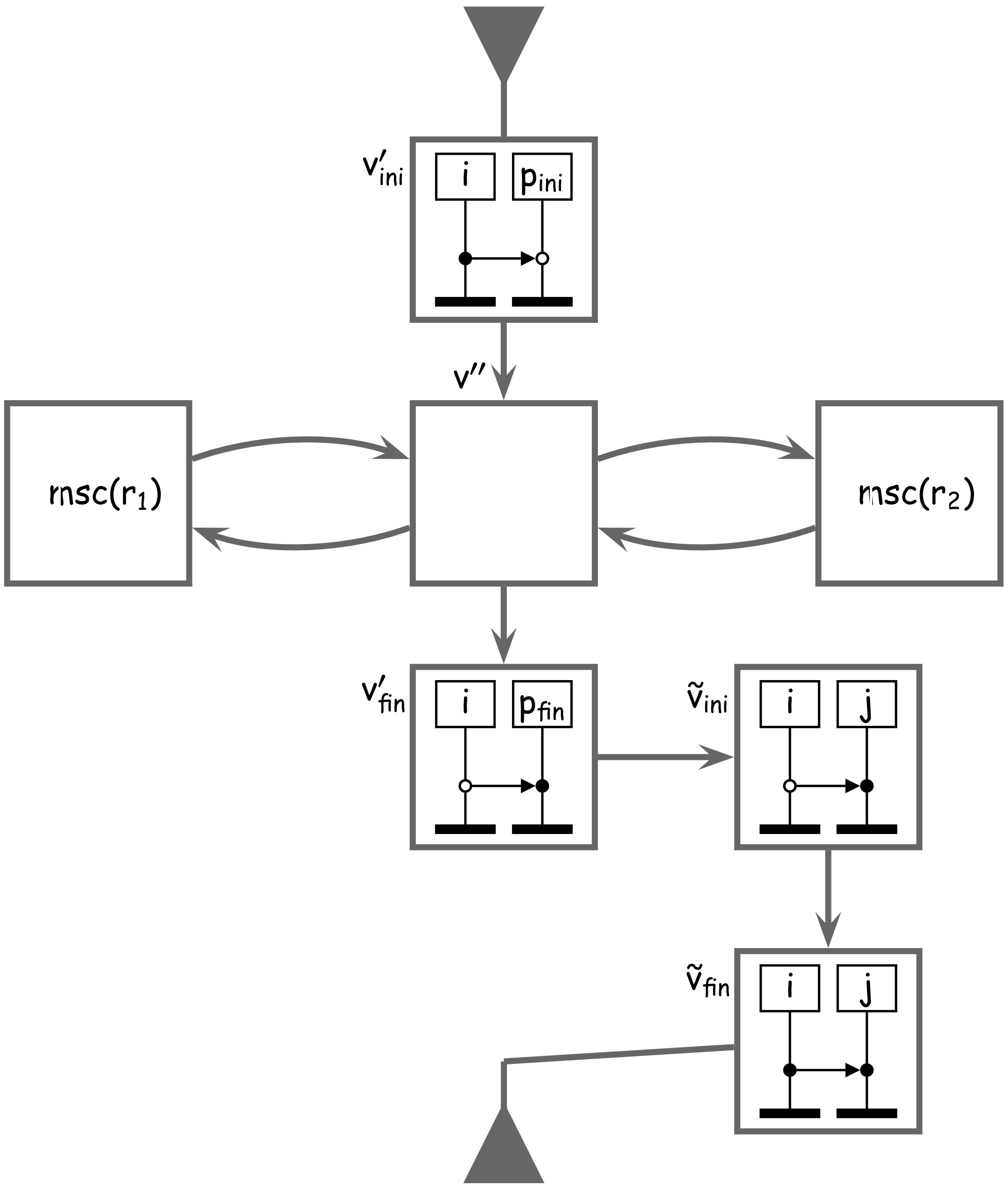
Remark

The emptiness problem for linear MSGs is EXPSPACE-complete.

Theorem

Checking the linearity of a given MSG is equivalent to Covering.

From Covering to Linearity



p_{fin} is covered from p_{ini}
if and only if G is not linear.

Characterization of undefined products

Let A and B be two (compositional) MSCs.

Proposition [Gunter et al, TACAS 2001]

The product $A \cdot B$ is defined if and only if the three next conditions are satisfied:

1. $^{\bullet}A = 0$, i.e. A is a prefix MSC
2. $A^{\bullet} \geqslant {}^{\bullet}B$, i.e. for each channel, the number of unmatched receives in B is at most equal to the number of unmatched sends in A.
3. For each channel c, if B contains a matching arrow in c then $A^{\bullet}(c) = {}^{\bullet}B(c)$, i.e. the number of unmatched receives in B is equal to the number of unmatched sends in A.

Checking linearity (1/3)

An MSG G is linear if and only if

for each vertex v

for each vertex v′ with $v' \to v$ in G

for each valid path from v_{ini} to v′ with product M

the product $M \cdot msc(v)$ is defined if and only if $M^\bullet \geqslant {}^\bullet msc(v)$.

Checking linearity (2/3)

An MSG G is linear if and only if

for each vertex v

for each channel c with a matching arrow in msc(v)

for each vertex v′ with $v' \to v$ in G

for each valid path from v_{ini} to v′ with product M

If $M^\bullet \geqslant {}^\bullet msc(v)$ then $M^\bullet(c) = {}^\bullet msc(v)(c)$.

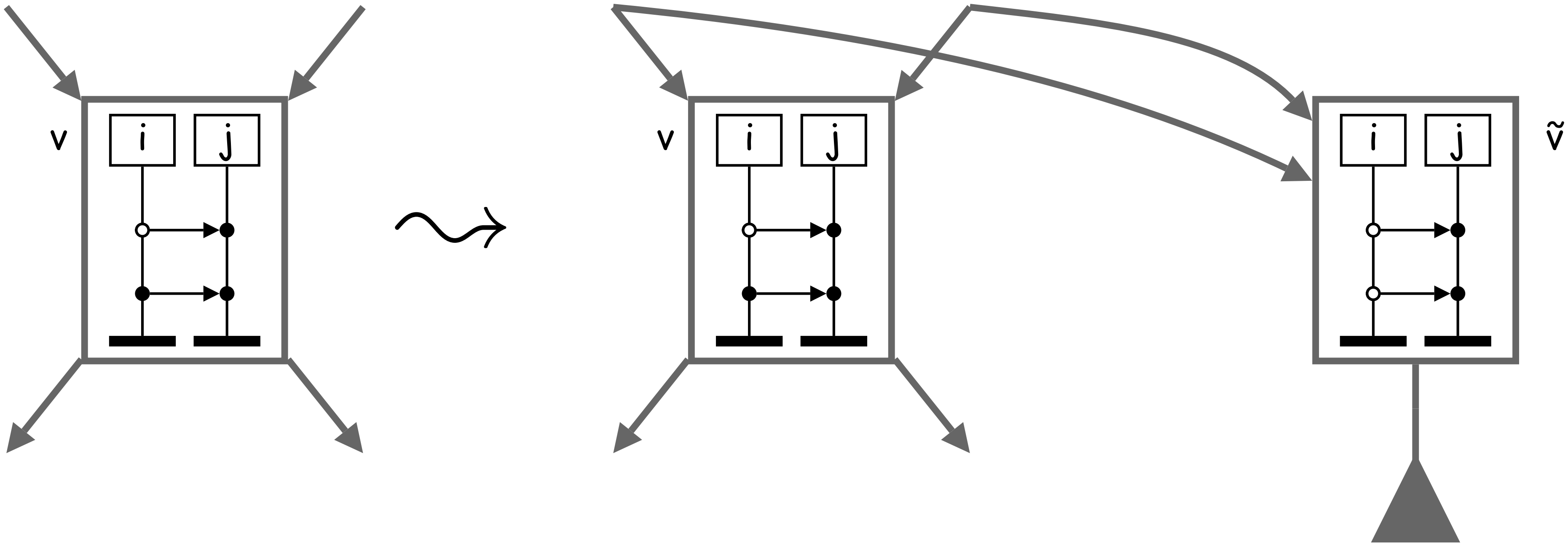
Checking linearity (3/3)

An MSG G is linear if and only if

for each vertex v

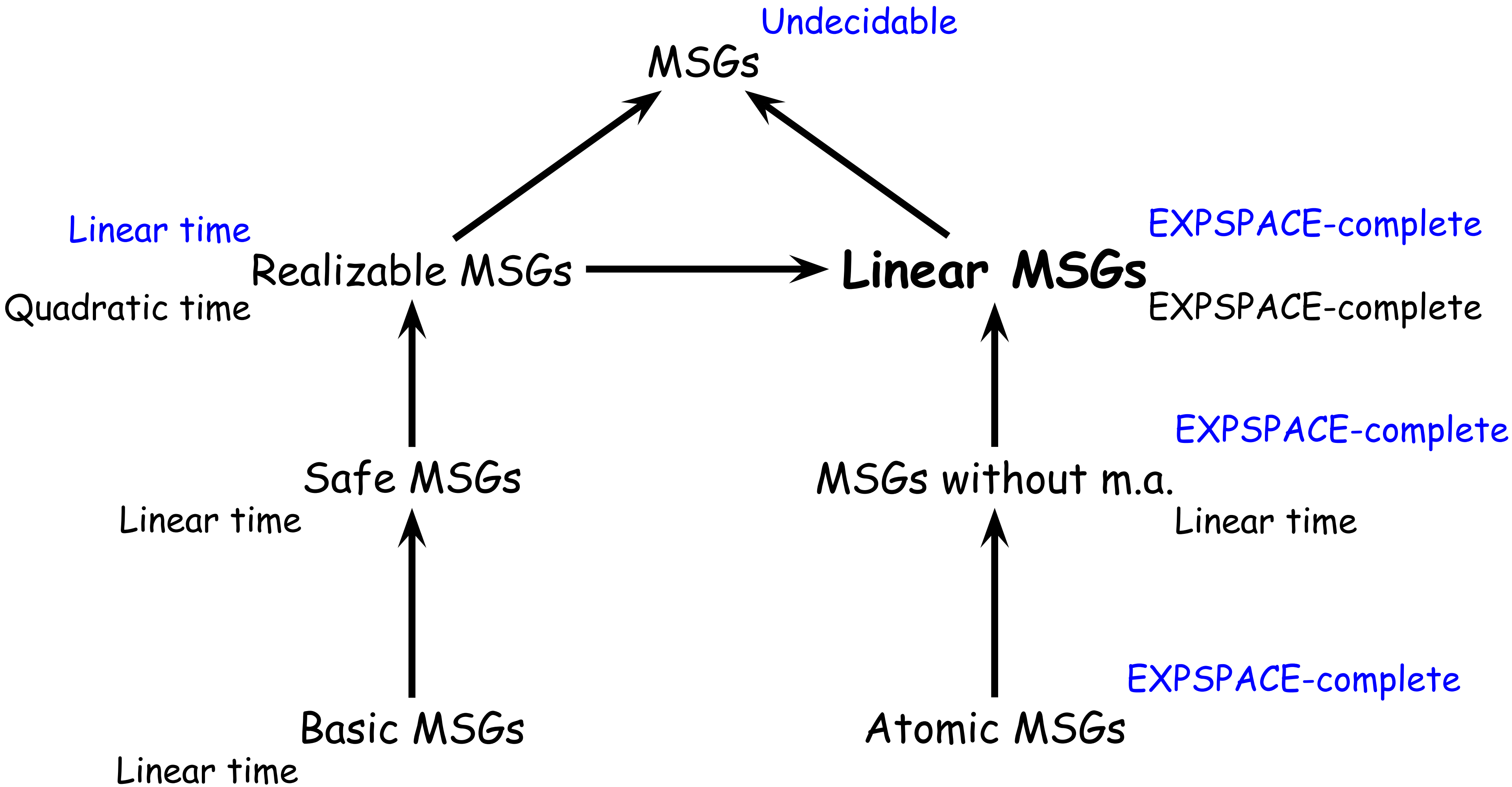
for each channel c with a matching arrow in $msc(v)$

$\mathcal{L}(G_{v,c}) = \emptyset$.



- ✔ Background
- ✔ Composition of compositional MSCs
- ✔ Emptiness is undecidable
- ✔ MSGs without matching arrow
- ✔ The subclass of linear MSGs
- ☛ Conclusion

Overview



Undecidable
MSGs
Linear time
Realizable MSGs
Quadratic time
EXPSPACE-complete
Linear MSGs
EXPSPACE-complete
EXPSPACE-complete
Safe MSGs
Linear time
MSGs without m.a.
Linear time
EXPSPACE-complete
Basic MSGs
Linear time
Atomic MSGs

Linear MSGs

- **Linearity is difficult to check** but some syntactic or behavioural restrictions guarantee linearity.
  1. Unmatched events are forbidden for channels with a matching arrow;
  2. Each channel with a matching arrow is "safe".
- Emptiness, boundedness and other **reachability properties** are decidable for linear MSGs and undecidable in general.

Linear MSGs with counters

- **Counters** can be added to the model with no difficulty.
- A linear MSG can be more difficult to check than a possible equivalent safe one, up to unfolding. But
  - MSGs with counters can be exponentially more **concise**.
  - MSGs with counters are easier to understand in practice.

From linear MSGs to atomic MSGs

- Each linear MSG can be transformed into an equivalent **atomic** one:

  $\Rightarrow$ Model checking bounded linear MSGs against MSO formulae is decidable;

  $\Rightarrow$ **Prefix-reachability properties**, such as universal boundedness, can be reduced to Petri nets, too;

  using results from **Avellaneda's thesis**.

- **Discrete timers** can be handled easily with atomic MSGs (and hence with linear MSGs).