

# Theoretical Foundations of the UML

## Lecture 14: Realising Local Choice MSGs

Joost-Pieter Katoen

Lehrstuhl für Informatik 2  
Software Modeling and Verification Group

`moves.rwth-aachen.de/teaching/ss-16/theoretical-foundations-of-the-uml/`

20. Juni 2016

- 1 Introduction
- 2 Local Choice MSGs
- 3 Regular Expressions over MSCs
- 4 A Realisation Algorithm for MSGs

- 1 Introduction
- 2 Local Choice MSGs
- 3 Regular Expressions over MSCs
- 4 A Realisation Algorithm for MSGs

## Definition (Realisability of MSGs)

- 1 MSG  $G$  is **realisable** whenever  $\mathcal{L}(G) = \mathcal{L}(\mathcal{A})$  for some CFM  $\mathcal{A}$ .
- 2 MSG  $G$  is **safely realisable** whenever  $\mathcal{L}(G) = \mathcal{L}(\mathcal{A})$  for some deadlock-free CFM  $\mathcal{A}$ .

## Results so far:

- ① Conditions for (safe) realisability for **finite** sets of MSCs.
- ② Checking these conditions is co-NP complete (in P).
- ③ Regular MSGs are (safely) realisable by  $\forall$ -bounded CFMs.
- ④ Checking regularity of MSGs is undecidable.
- ⑤ Communication-closedness implies regularity; its check is co-NP complete.
- ⑥ Local communication-closedness implies regularity, and can be checked in P.

## Some remaining questions

- Can results be obtained for **larger classes** of MSGs?
- What happens if we allow **synchronisation messages**?
  - recall that weak CFMs do not involve synchronisation messages
- How do we obtain a CFM realising an MSG **algorithmically**?
  - in particular, for non-local choice MSGs

## Today's lecture

Safe realisability of (a somewhat restricted class of) MSGs. So as to obtain deadlock-free CFMs, the input MSG is required to be **local choice**. The CFM are **not** required to be weak. The algorithm will exploit synchronisation messages.

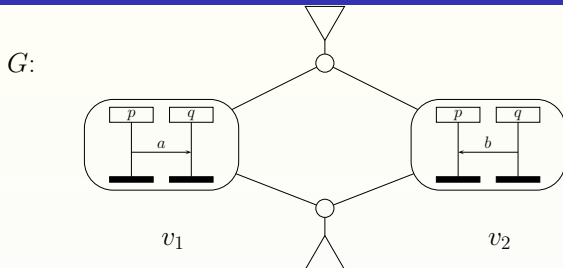
## Results:

- 1 Realisability for constrained regular expressions of local-choice MSGs.
- 2 An algorithm that generates a CFM from such local-choice MSG.

- 1 Introduction
- 2 Local Choice MSGs
- 3 Regular Expressions over MSCs
- 4 A Realisation Algorithm for MSGs



# Non-local choice



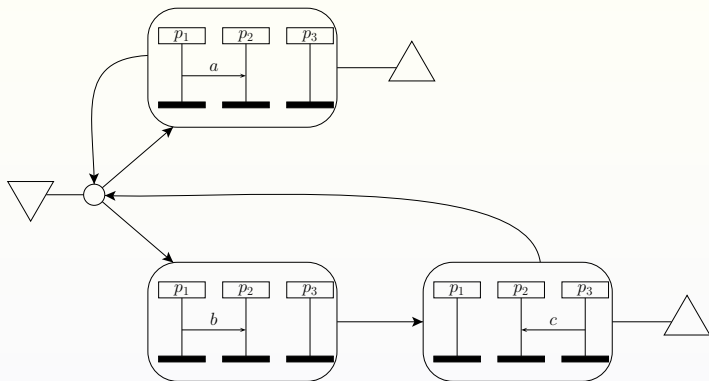
**Inconsistency** if process  $p$  behaves according to vertex  $v_1$  and process  $q$  behaves according to vertex  $v_2$

$\implies$  realisation by a CFM may yield a deadlock

## Problem:

Subsequent behavior in  $G$  is determined by **distinct** processes. When several processes independently decide to initiate behavior, they might start executing different successor MSCs (= vertices). This is called a **non-local choice**.

## A (more involved) non-local choice



### Problem:

Inconsistency if  $p_1$  decides to send  $a$  and  $p_3$  decides to send  $c$ .  
Which branch to take in the initial vertex?

## Definition (Minimal event)

Let  $(E, \preceq)$  be a poset. Event  $e \in E$  is a **minimal** event wrt.  $\preceq$  if  $\neg(\exists e' \neq e. e' \preceq e)$ .

Intuition: there is no event that has to happen before  $e$  happens.  
That is to say: the occurrence of  $e$  does not depend on any other event.

## Definition (Partial order of a path)

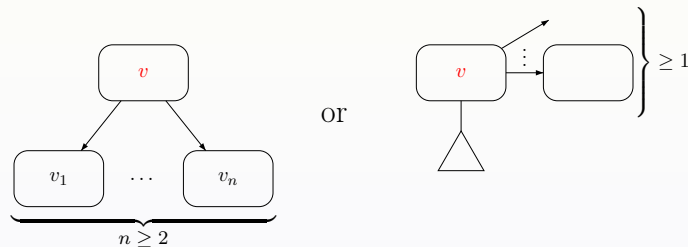
For finite path  $\pi = v_1 \dots v_n$  in MSG  $G$ , let  $<_{M(\pi)}$  be the partial order of the MSC  $M(\pi) = \lambda(v_1) \bullet \dots \bullet \lambda(v_n)$ .

Let  $\min(\pi)$  be the **set of minimal events** wrt.  $<_{M(\pi)}$  along finite path  $\pi$ .

# Branching vertices

A **branching** vertex in MSG  $G$  either has at least two successors, or is a final vertex with at least one successor.

Pictorially, vertex  $v$  is **branching** if either:



Without loss of generality we assume that branching final vertices do not occur. They can be always be removed at the expense of copying such vertices.

# Local choice property

## Definition (Local choice)

Let MSG  $G = (V, \rightarrow, v_0, F, \lambda)$ . MSG  $G$  is **local choice** if for every branching vertex  $v \in V$  it holds:

$$\exists \text{process } p. (\forall \pi \in \text{Paths}(v). |\min(\pi')| = 1 \wedge \min(\pi') \subseteq E_p)$$

where for  $\pi = vv_1v_2 \dots v_n$  we have  $\pi' = v_1v_2 \dots v_n$ .

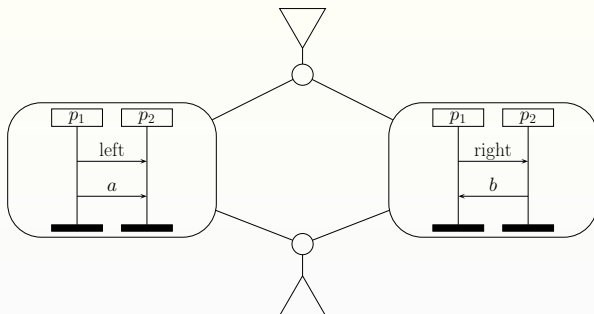
## Intuition:

There is a single process that initiates behavior along every path from the branching vertex  $v$ . This process decides how to proceed. In a realisation by a CFM, it can inform the other processes how to proceed.

## Local choice or not?

Deciding whether MSG  $G$  is local choice or not is in P. (Exercise.)

$G$ :



How to resolve a non-local choice?

Amend your MSG and add control messages (cf. above example)

- 1 Introduction
- 2 Local Choice MSGs
- 3 Regular Expressions over MSCs**
- 4 A Realisation Algorithm for MSGs

## Definition (Asynchronous iteration)

For  $\mathcal{M}_1, \mathcal{M}_2 \subseteq \mathbb{M}$  sets of MSCs, let:

$$\mathcal{M}_1 \bullet \mathcal{M}_2 = \{ M_1 \bullet M_2 \mid M_1 \in \mathcal{M}_1, M_2 \in \mathcal{M}_2 \}$$

For  $\mathcal{M} \subseteq \mathbb{M}$  let

$$\mathcal{M}^i = \begin{cases} \{M_\epsilon\} & \text{if } i=0, \text{ where } M_\epsilon \text{ denotes the empty MSC} \\ \mathcal{M} \bullet \mathcal{M}^{i-1} & \text{if } i > 0 \end{cases}$$

The **asynchronous iteration** of  $\mathcal{M}$  is now defined by:

$$\mathcal{M}^* = \bigcup_{i \geq 0} \mathcal{M}^i.$$



## Definition (Regular expressions over MSCs)

The set  $\text{REX}_{\mathbb{M}}$  of **regular expressions** over  $\mathbb{M}$  is given by the grammar:

$$\alpha ::= \emptyset \mid M \mid \alpha_1 \cdot \alpha_2 \mid \alpha_1 + \alpha_2 \mid \alpha^*$$

where MSC  $M \in \mathbb{M}$ .

## Definition (Semantics of regular expressions, $\mathcal{L}(\cdot) : \text{REX}_{\mathbb{M}} \rightarrow 2^{\mathbb{M}}$ )

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(M) = \{M\}$
- $\mathcal{L}(\alpha_1 \cdot \alpha_2) = \mathcal{L}(\alpha_1) \bullet \mathcal{L}(\alpha_2)$
- $\mathcal{L}(\alpha_1 + \alpha_2) = \mathcal{L}(\alpha_1) \cup \mathcal{L}(\alpha_2)$
- $\mathcal{L}(\alpha^*) = \mathcal{L}(\alpha)^*$

## Definition (Locally accepting CFM)

CFM  $\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$  is **locally accepting** (la, for short) if

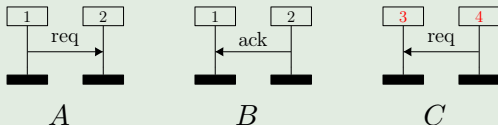
$$F = \prod_{p \in \mathcal{P}} F_p \quad \text{where} \quad F_p \subseteq S_p.$$

Thus: every combination of local accept states is a global accept state of the CFM.

# Regular expressions for MSCs

Let  $\mathcal{P} = \{1, 2, 3, 4\}$  and  $\mathcal{C} = \{\text{req}, \text{ack}\}$ .

## Example



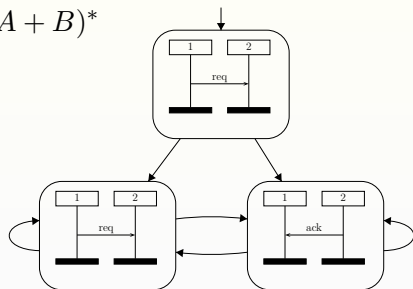
Consider the following regular expressions over  $\mathbb{M}$ :

- $\alpha_1 = (A \cdot B)^*$       det.  $\forall 1$ -bounded deadlock-free weak la CFM
- $\alpha_2 = (A + B)^*$       det.  $\exists 1$ -bounded la CFM
- $\alpha_3 = (A \cdot C)^*$       not realisable
- $\alpha_4 = A \cdot (A + B)^*$        $\exists 1$ -bounded deadlock-free la CFM

How about realisability of  $\mathcal{L}(\alpha_i)$ ?

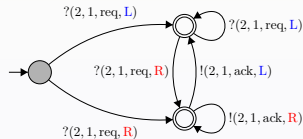
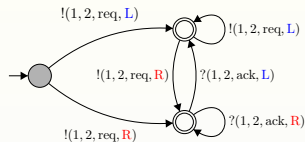
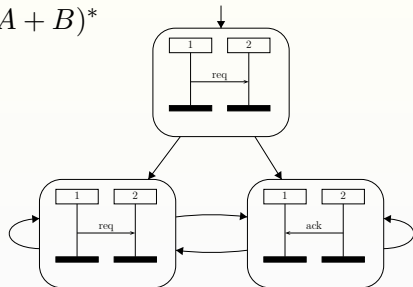
# Realising local-choice expressions by deadlock-free CFMs

$$A \cdot (A + B)^*$$



# Realising local-choice expressions by deadlock-free CFMs

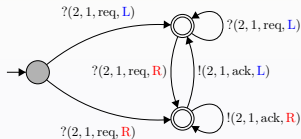
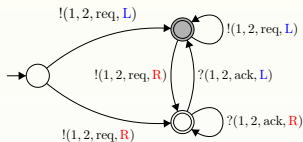
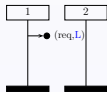
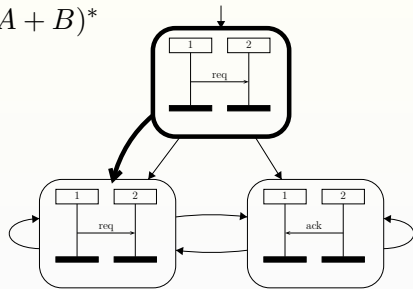
$$A \cdot (A + B)^*$$



1 → 2 :
2 → 1 :

# Realising local-choice expressions by deadlock-free CFMs

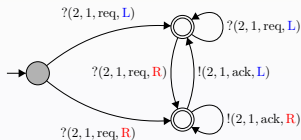
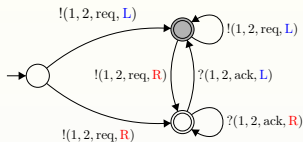
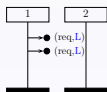
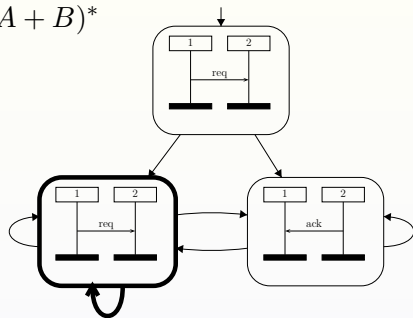
$$A \cdot (A + B)^*$$



1 → 2 : (req,L)  
2 → 1 :

# Realising local-choice expressions by deadlock-free CFMs

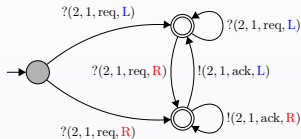
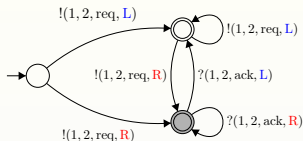
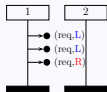
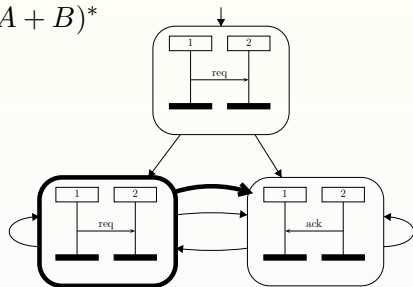
$$A \cdot (A + B)^*$$



$1 \rightarrow 2 : (\text{req},L) (\text{req},L)$ $2 \rightarrow 1 :$
--

# Realising local-choice expressions by deadlock-free CFMs

$$A \cdot (A + B)^*$$

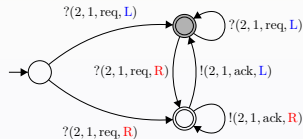
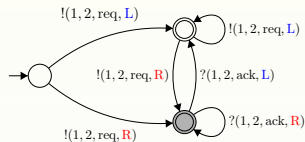
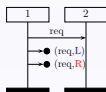
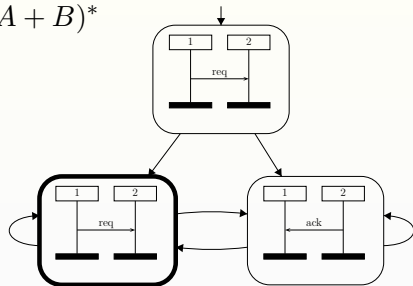


1 → 2 :	(req,L)	(req,L)	(req,R)
2 → 1 :			



# Realising local-choice expressions by deadlock-free CFMs

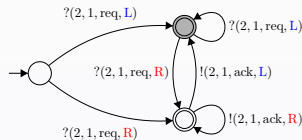
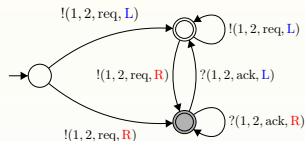
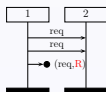
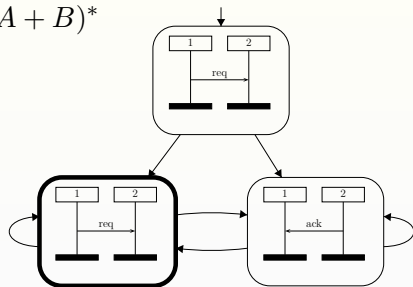
$$A \cdot (A + B)^*$$



$1 \rightarrow 2 : (req, L) (req, R)$   
 $2 \rightarrow 1 :$

# Realising local-choice expressions by deadlock-free CFMs

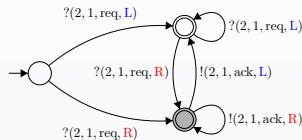
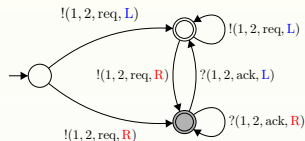
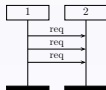
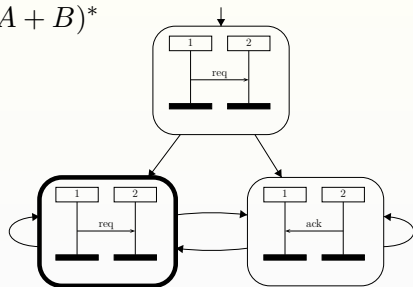
$$A \cdot (A + B)^*$$



1 → 2 : (req,R)  
2 → 1 :

# Realising local-choice expressions by deadlock-free CFMs

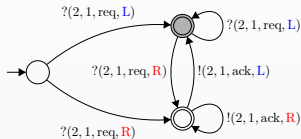
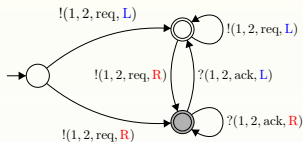
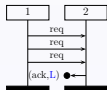
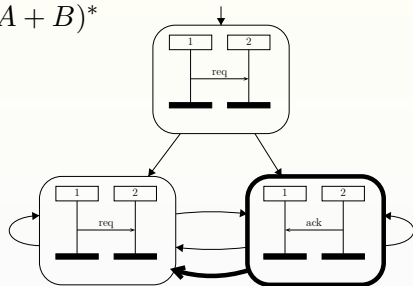
$$A \cdot (A + B)^*$$



1 → 2 :  
2 → 1 :

# Realising local-choice expressions by deadlock-free CFMs

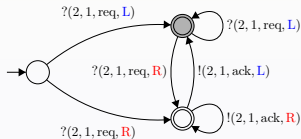
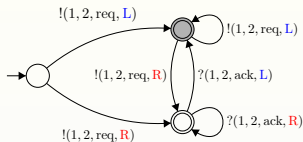
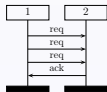
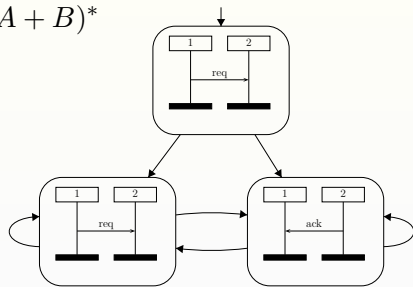
$$A \cdot (A + B)^*$$



1 → 2 :  
2 → 1 : (ack,L)

# Realising local-choice expressions by deadlock-free CFMs

$$A \cdot (A + B)^*$$



1 → 2 :  
2 → 1 :

## Definition (Connected MSC)

An MSC  $M = (\mathcal{P}, E, \mathcal{C}, l, m, <) \in \mathbb{M}$  is **connected** if:

$$\forall e, e' \in E. (e, e') \in (< \cup <^{-1})^*$$

## Definition (Star-connected)

Regular expression  $\alpha \in \text{REX}_{\mathbb{M}}$  is **star-connected** if, for any subexpression  $\beta^*$  of  $\alpha$ ,  $\mathcal{L}(\beta)$  is a set of connected MSCs.

Examples on the black board.

## Definition (Finitely generated)

Set of MSCs  $\mathcal{M} \subseteq \mathbb{M}$  is **finitely generated** if there is a finite set of MSCs  $\widehat{\mathcal{M}} \subseteq \mathbb{M}$  such that  $\mathcal{M} \subseteq \widehat{\mathcal{M}}^*$ .

## Theorem

[Morin 2002]

Let  $\mathcal{M}$  be finitely generated. Then:

$\mathcal{M}$  is realisable

iff

there exists a **star-connected** regular expression  $\alpha$  with  $\mathcal{L}(\alpha) = \mathcal{M}$ .

- 1 Introduction
- 2 Local Choice MSGs
- 3 Regular Expressions over MSCs
- 4 A Realisation Algorithm for MSGs**



An example local-choice MSG on black board.

## Theorem

[Genest *et al.*, 2005]

Any local-choice MSG  $G$  is safely realisable by a CFM with additional synchronisation data (which is of size linear in  $G$ ).

## Proof

As MSG  $G$  is local choice, at every branch  $v$  of  $G$  there is a unique process,  $p(v)$ , say, such that on every path from  $v$  the unique minimal event occur at  $p(v)$ . Then:

- 1 Process  $p(v)$  determines the successor vertex of  $v$ .
- 2 Process  $p(v)$  informs all other processes about its decision by adding synchronisation data to the exchanged messages.
- 3 Synchronisation data is the path (in  $G$ ) from  $v$  to the next branching vertex along the direction chosen by  $p(v)$ .

# Maximal non-branching paths

## Definition (Maximal non-branching paths)

For MSG  $G = (V, \rightarrow, v_0, F, \lambda)$ , let  $nbp : V \rightarrow V^*$  be defined by:

$$nbp(v) = \begin{cases} v & \text{if } v \in F \text{ or } v \text{ is a branching vertex} \\ v_1 \dots v_n & \text{otherwise} \end{cases}$$

where  $v_1 \dots v_n \in V^*$  is a maximal path (i.e., a path that cannot be prolonged) satisfying:

- 1  $v_i = v$  for some  $i$ ,  $0 < i \leq n$ , and
- 2  $v_n \in F$  or is a branching vertex, and
- 3  $v_1 = v_0$  or is a direct successor of a branching vertex, and
- 4  $v_2, \dots, v_{n-1} \notin F$  and are all non-branching vertices

## Intuition

$nbp(v)$  is the **maximal non-branching path** to which  $v$  belongs.

# Structure of the CFM of local choice MSG $G$

Let MSG  $G = (V, \rightarrow, v_0, F, \lambda)$  be local choice.

Define the CFM  $\mathcal{A}_G = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F')$  with:

- 1 Local automaton  $\mathcal{A}_p = (S_p, \Delta_p)$  as defined on next slides
- 2  $\mathbb{D} = \{ npb(v) \mid v \in V \}$   
synchronisation data = maximal non-branching paths in  $G$
- 3  $s_{init} = \{ (v_0, \emptyset) \}^n$  where  $n = |\mathcal{P}|$   
each local automaton  $\mathcal{A}_p$  starts in initial state  $(v_0, \emptyset)$ , i.e., in initial vertex  $v_0$  while no events of  $p$  have been performed
- 4  $\bar{s} \in F'$  iff for all  $p \in \mathcal{P}$ , local state  $\bar{s}[p] = (v, E)$  with  $E \subseteq E_p$  and:
  - 1  $v \in F$  and  $E$  contains a maximal event wrt.  $<_p$  in MSC  $\lambda(v)$ , or
  - 2  $v \notin F$  and  $\pi = v \dots w$  is a path in  $G$  with  $w \in F$  and  $E$  contains a maximal event wrt.  $<_p$  in MSC  $\lambda(\pi)$ .

- $S_p = V \times E_p$  such that for any  $s = (v, E) \in S_p$ :

$$\forall e, e' \in \lambda(v). (e <_p e' \text{ and } e' \in E \text{ implies } e \in E)$$

that is,  $E$  is downward-closed with respect to  $<_p$  in MSC  $\lambda(v)$

- Intuition: a state  $(v, E)$  means that process  $p$  is currently in vertex  $v$  of MSG  $G$  and has already performed the events  $E$  of  $\lambda(v)$
- Initial state of  $\mathcal{A}_p$  is  $(v_0, \emptyset)$

- Executing events **within a vertex** of the MSG  $G$ :

$$\frac{e \in E_p \cap \lambda(v) \text{ and } e \notin E}{(v, E) \xrightarrow{l(e), nbp(v)}_p (v, E \cup \{e\})}$$

Note: since  $E \cup \{e\}$  is downward-closed wrt.  $<_p$ ,  $e$  is enabled

- **Taking an edge** (possibly a self-loop) of the MSG  $G$ :

$$\frac{E = E_p \cap \lambda(v) \text{ and } e \in E_p \cap \lambda(w) \text{ and } vu_0 \dots u_n w \in V^* \text{ with } p \text{ not active in } u_0 \dots u_n}{(v, E) \xrightarrow{l(e), nbp(w)}_p (w, \{e\})}$$

Note: vertex  $w$  is the first successor vertex of  $v$  on which  $p$  is active

A couple of examples on the black board.