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-20/fuml/

June 9, 2020

Outline

Overview

Today's topic

Today's lecture

An algorithm to realise local-choice MSGs using CFMs with synchronisation messages.

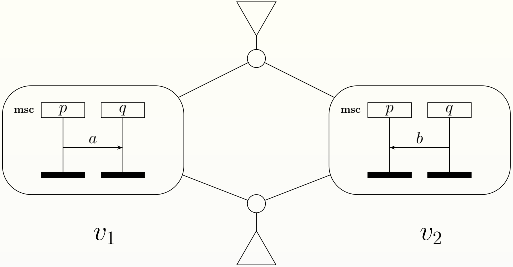
Results:

1. An algorithm that generates a CFM from local-choice MSGs.

Overview

Non-local choice

G:



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

$\Longrightarrow$ 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.

Local choice property

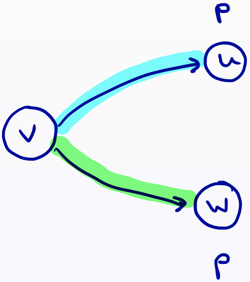
Definition (Local choice)

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

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

where for $\pi = v v_1 v_2 \ldots v_n$ we have $\pi' = v_1 v_2 \ldots v_n$.

MSG G:



p

$\pi_1 = v u \quad \pi_1' = u$

$\pi_2 = v w \quad \pi_2' = w$

Local choice property

Definition (Local choice)

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

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

where for $\pi = v v_1 v_2 \ldots v_n$ we have $\pi' = v_1 v_2 \ldots v_n$.

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.

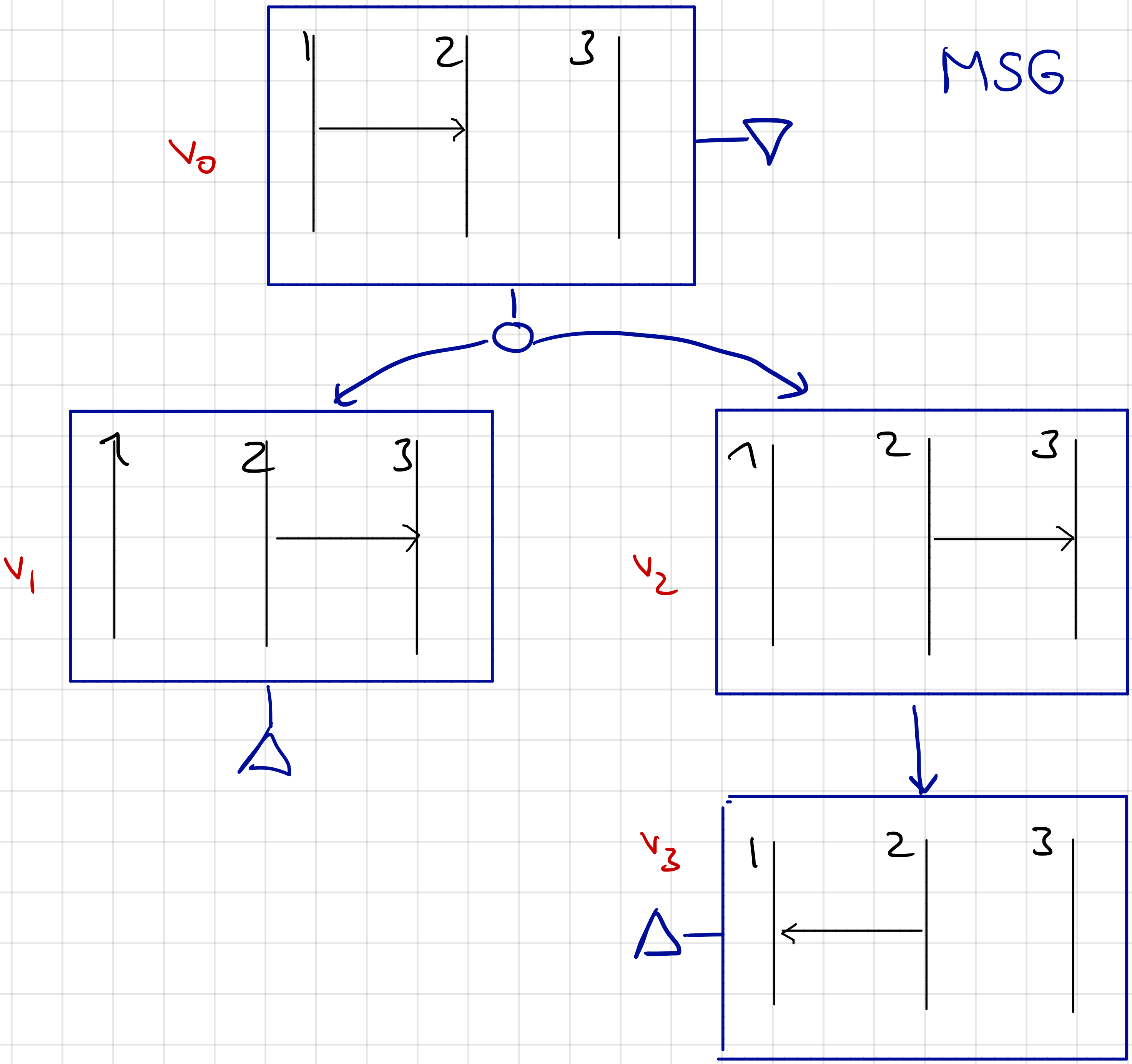
Local choice or not?

Deciding whether MSG G is local choice or not is in P.

Overview

Local choice MSGs

An example local-choice MSG on black board.



MSG
v0
1
2
3
v1
1
2
3
v2
1
2
3
v3
1
2
3

Realising local choice (C)MSGs

Theorem [Genest *et al.*, 2005]

Any local-choice MSG G is safely realisable by a CFM with 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 successor vertex (in G) from v chosen by $p(v)$.

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

pairs (V, E)

downward-closed wrt $<_p$

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. $\mathbb{D} \; = \; V$

   synchronisation data = vertices in the MSG

   every process starts in the initial vertex with no events executed so far
3. $s_{init} = \{ \, (v_0, \varnothing) \, \}^n$ where $n = |\mathcal{P}|$

   each local automaton $\mathcal{A}_p$ starts in initial state $(v_0, \varnothing)$, i.e., in initial vertex v_0 while no events of p have been performed

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

   slide 13 + 14

2. $\mathbb{D} = V$

   synchronisation data = vertices in the MSG

3. $s_{init} = \{ (v_0, \varnothing) \}^n$ where $n = |\mathcal{P}|$

   each local automaton $\mathcal{A}_p$ starts in initial state $(v_0, \varnothing)$, i.e., in initial vertex v_0 while no events of p have been performed

   $(s_1, s_2, \ldots, s_n)$ $|P| = n$

4. $\overline{s} \in F'$ iff for all $p \in \mathcal{P}$, local state $\overline{s}[p] = (v, E)$ 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 $\pi = v \ldots w$ is a path in G with $w \in F$ and E contains a maximal event wrt. $<_p$ in MSC $M(\pi)$.

State space of local automaton $\mathcal{A}_p$

vertex in the MSG

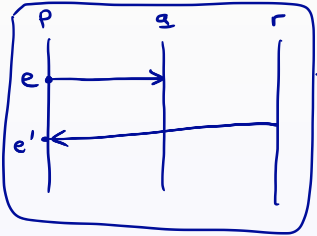
E_p

the set of events that process p has executed so far in vertex v $\lambda(v)$

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

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

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



vertex v

$(v, \{e', e\})$

State space of local automaton $\mathcal{A}_p$

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

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

  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 $(v_0, \varnothing)$

Transition relation of local automaton $\mathcal{A}_p$

- Executing events within a vertex of the MSG G:

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

event occurs at process p in vertex v and has not occured already

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

current state of A_p

$$(v, E) \xrightarrow{l(e),\, v} (v, E')$$

$\in \mathbb{D}$

$E \cup \{e\}$

Transition relation of local automaton $\mathcal{A}_p$

- Executing events within a vertex of the MSG G:

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

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

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

$$\frac{\begin{array}{c} E = E_p \cap \lambda(v) \text{ and } e \in E_p \cap \lambda(w) \text{ and} \\ v u_0 \ldots u_n w \in V^* \text{ with } p \text{ not active in } u_0 \ldots u_n \end{array}}{(v, E) \xrightarrow{l(e),w}_p (w, \{e\})}$$

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

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

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

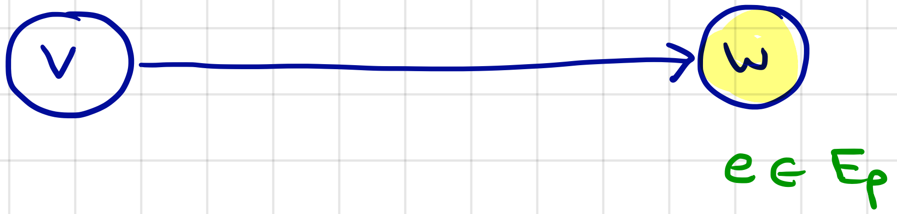
(*)

only event executed by p in vertex w

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

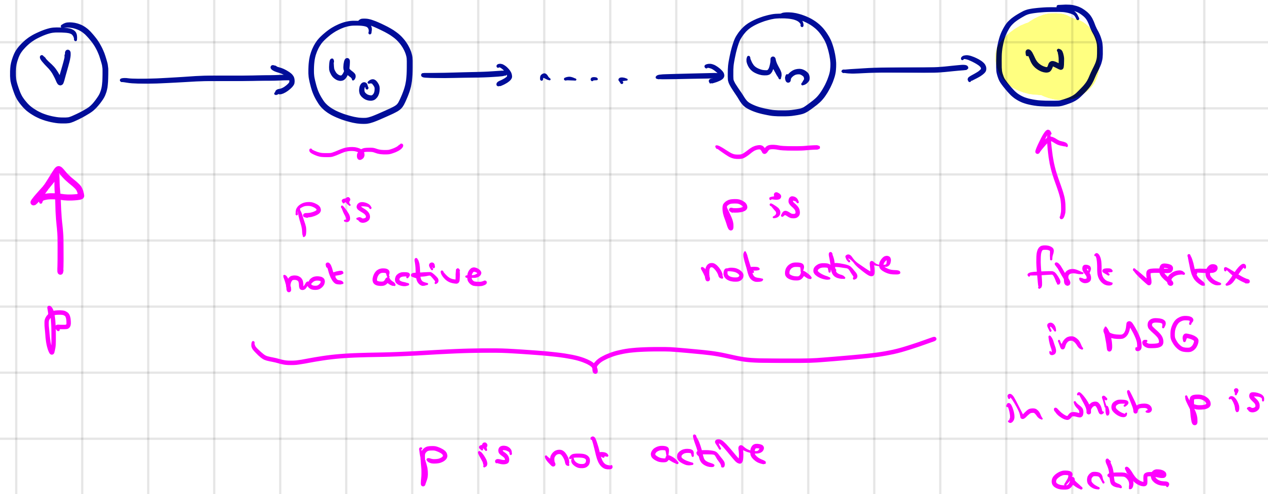
all events that p executed "in" vertex v

① MSG

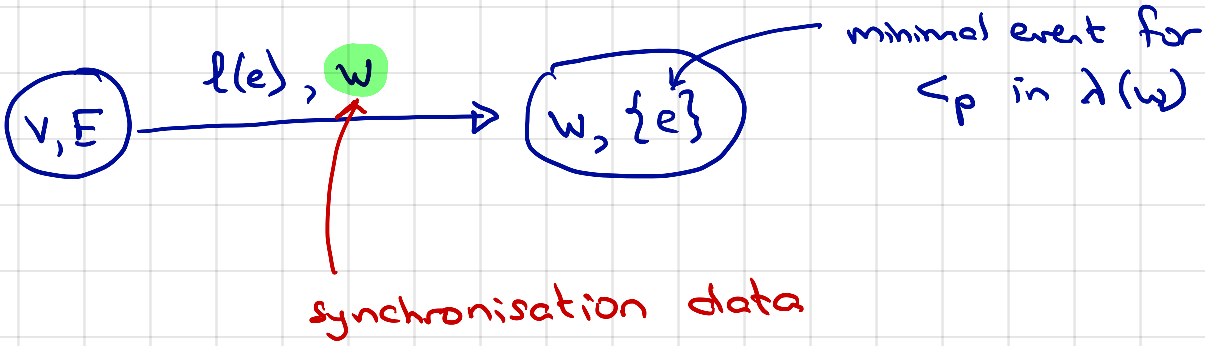


② MSG

(*)

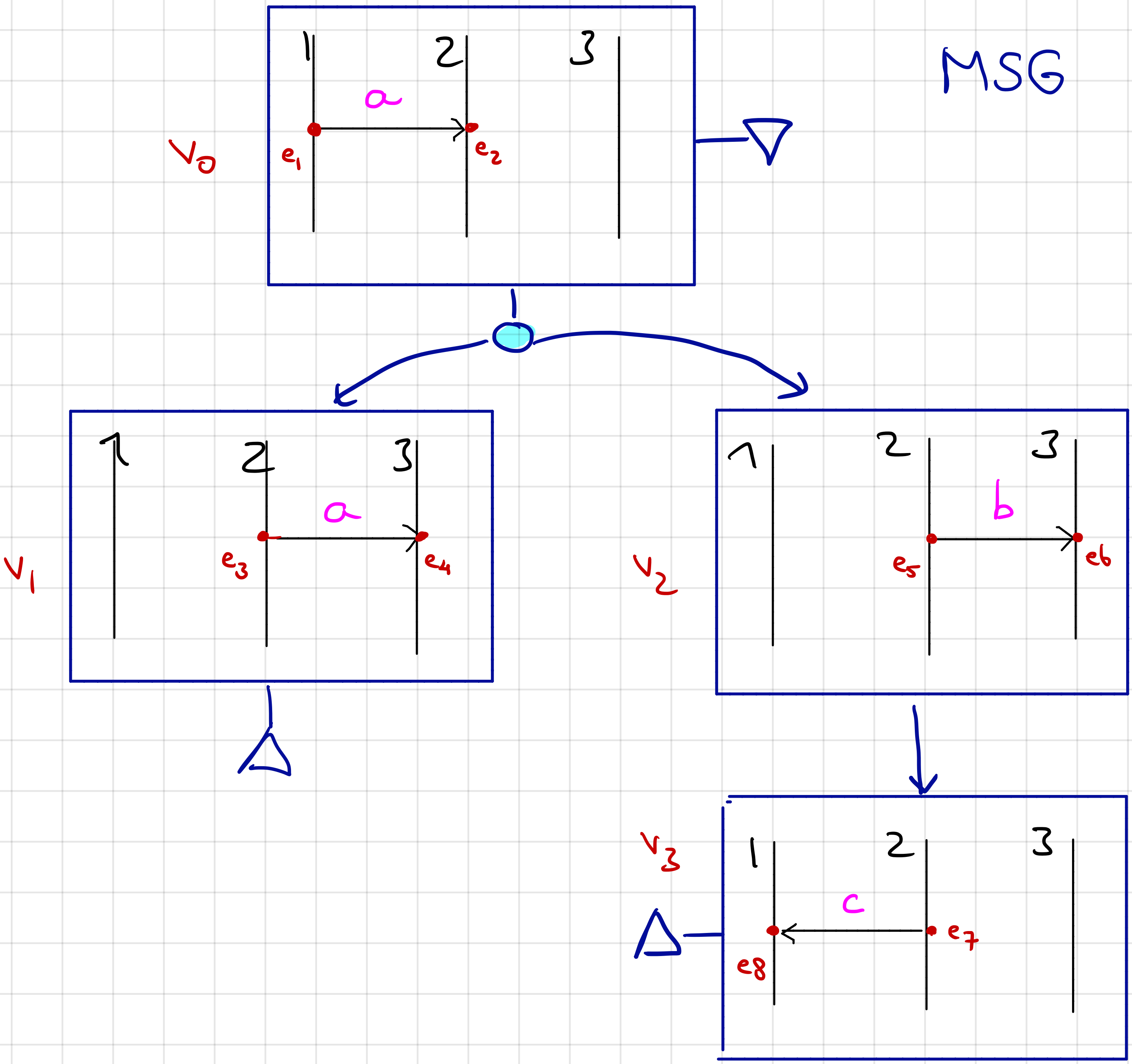


A_p



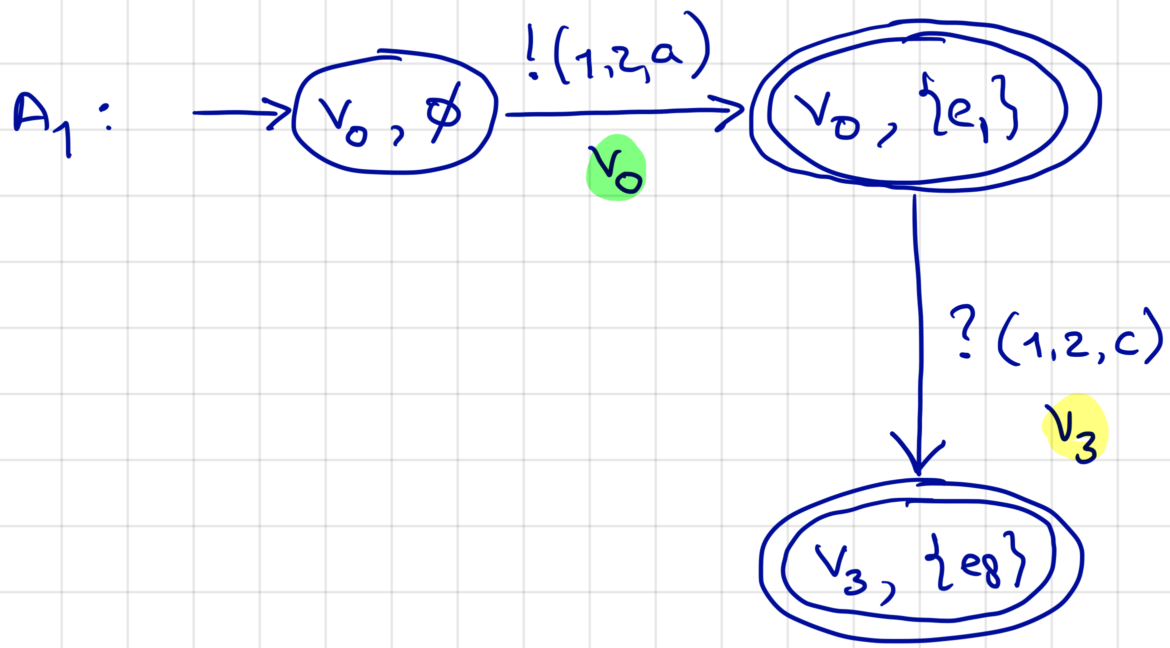
Examples

On the black board.



MSG
V0
1
2
3
a
e1
e2
V1
1
2
3
a
e3
e4
V2
1
2
3
b
e5
e6
V3
1
2
3
c
e7
e8

Local automata A_1, A_2, A_3



$\{e_1\}$

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

v_0 e_8 v_3

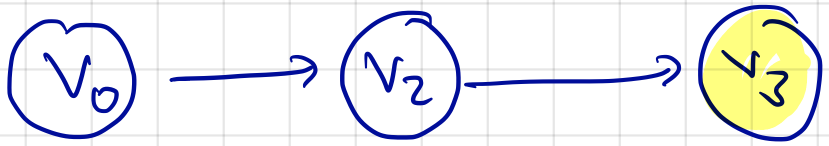
- Taking an edge (possibly a self-loop) of the MSG G:

$$\frac{E = E_p \cap \lambda(v) \text{ and } e \in E_p \cap \lambda(w) \text{ and } \quad \underbrace{vu_0 \ldots u_n}_{v_2} w \in V^* \text{ with } p \text{ not active in } \cancel{u_0 \ldots u_n}}{(v, E) \xrightarrow{l(e),w}_p (w, \{e\})}$$

v_2

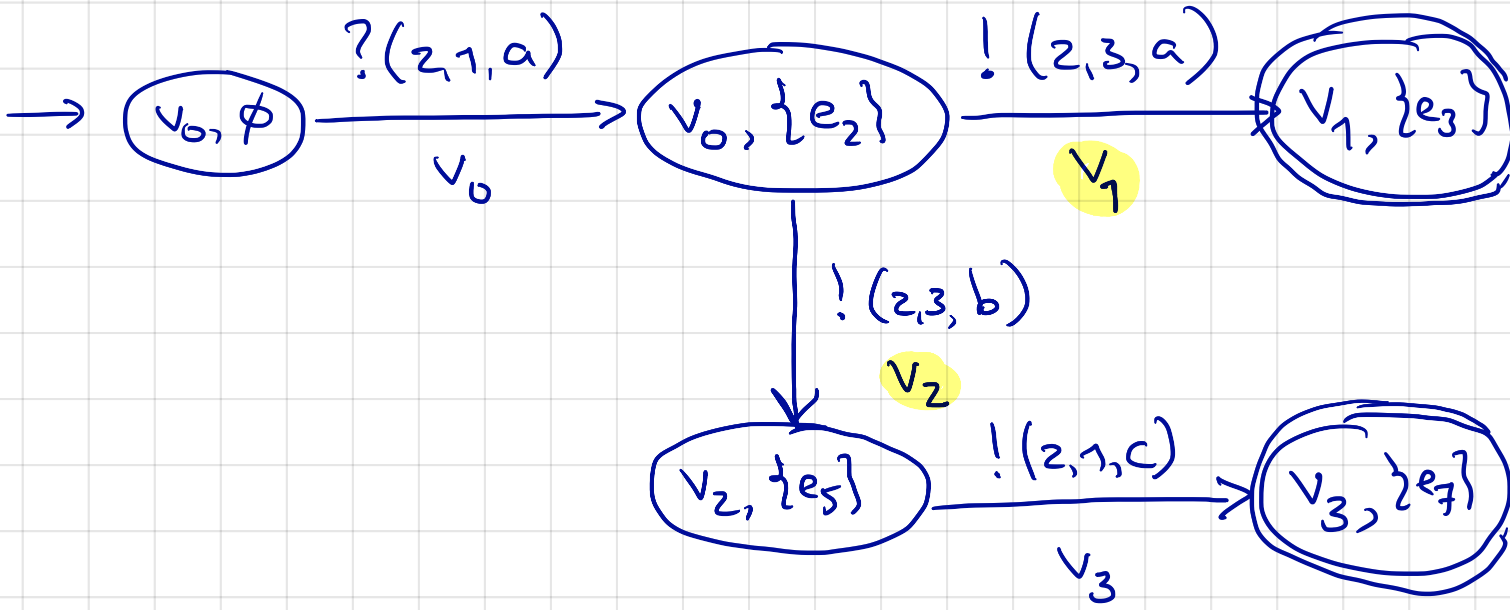
v_3

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



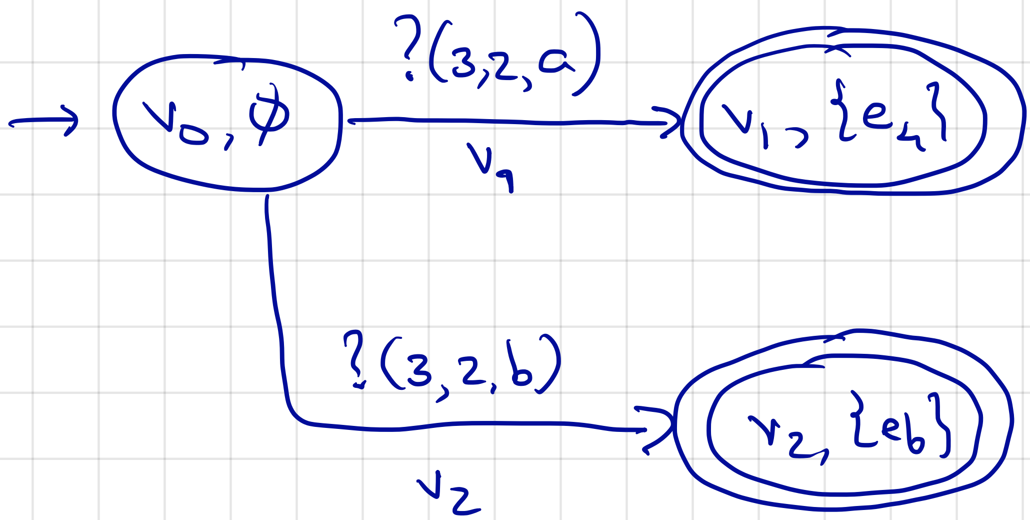
← process 1 is inactive

local automaton A_2

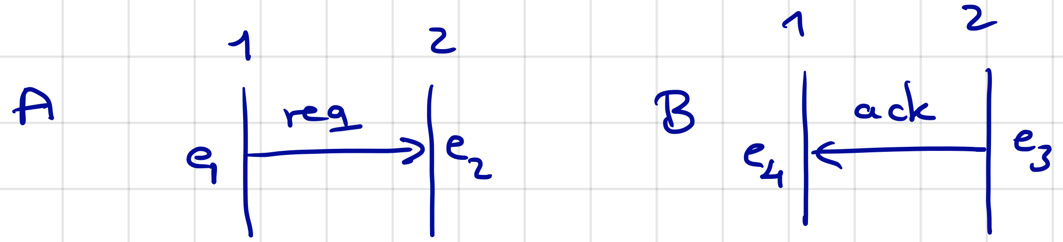


$$\frac{e \in E_p \cap \lambda(v) \text{ and } e \notin E}{(v, E) \xrightarrow{l(e),v}_p (v, E \cup \{e\})}$$

local automaton A_3

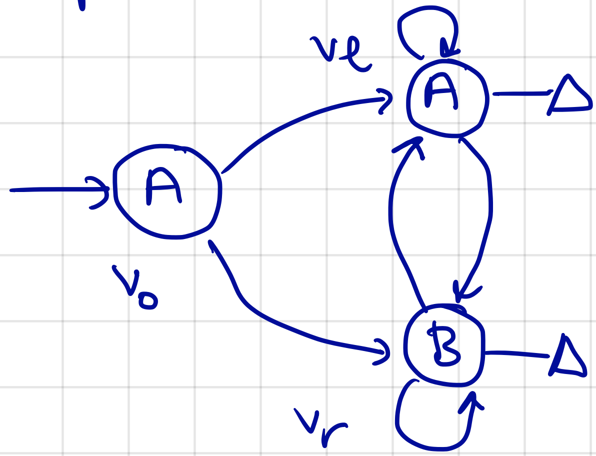


Second example



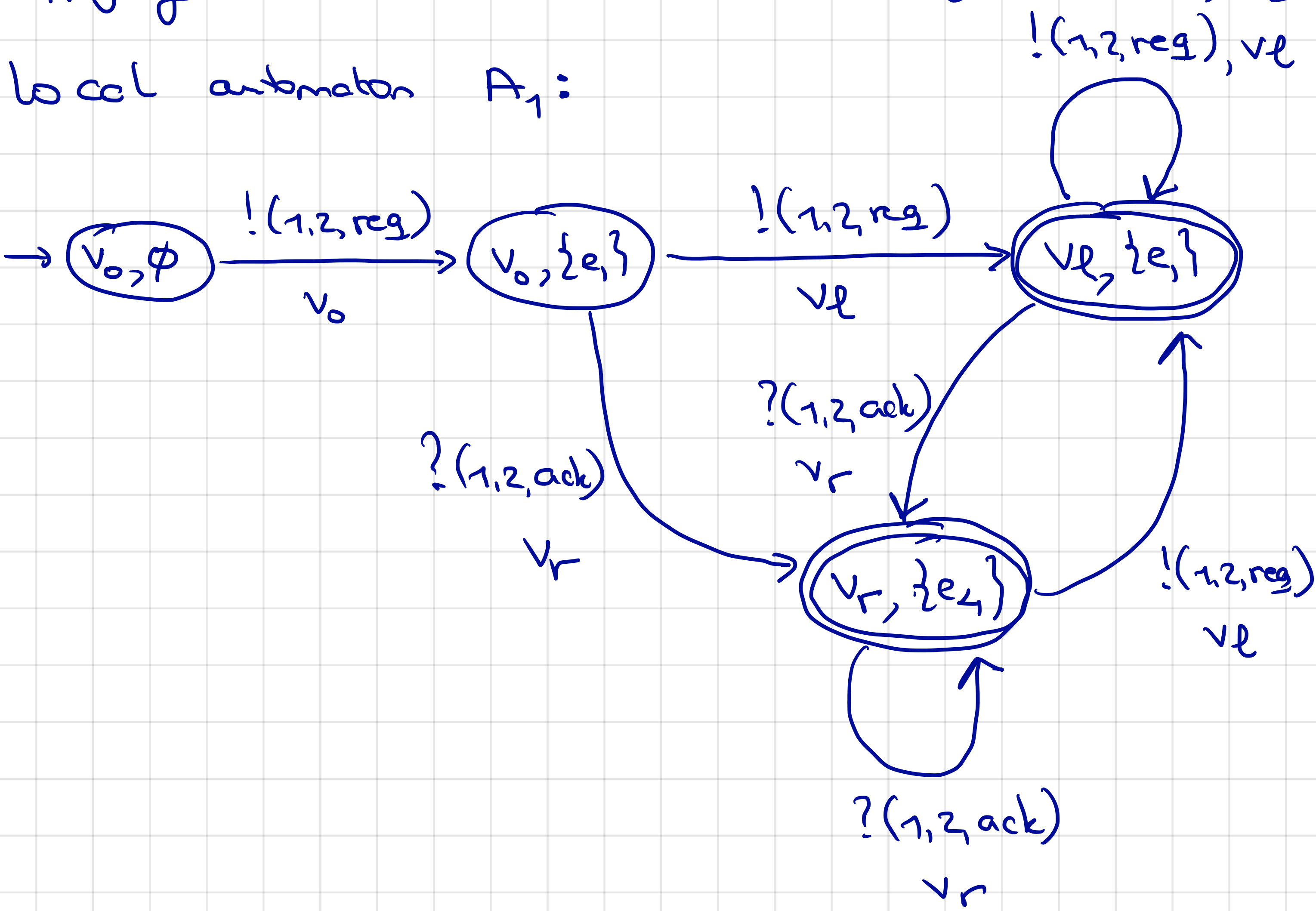
regular expression $= A \cdot (A+B)^*$

MSG



Applying the realisation construction yields A_1, A_2

local automaton A_1:



local automaton A_2: