Software Modeling and Verification
Lehrstuhl für Informatik 2
RWTH Aachen University

Prof. Dr. Ir. J.-P. Katoen
C. Kern

Foundations of the UML

Winter Term 07/08

– Lecture number 2: Message Sequence Graphs –

(2007-11-05)
summarized by *Thomas Bille* and *Tim Kosse*

1 Preliminaries from lecture 1

Sequence Diagramms. *Sequence diagrams specify the interaction patterns between the system components and are a popular elicitation technique for requirements engineering.*

Communication Action. *Let P be finite set of ≥ 2 sequential processes and $\mathbb{C}$ be a finite set of message contents $a, b, c \in \mathbb{C}$ then communication actions p, $q \in P$, $p \neq q$, $a \in \mathbb{C}$ are*

p ! q (a) *"p sends a message to q"*
p ? q (a) *"p receives a message sent by q"*

Partial Order. *For E a set of events, a partial order over E is a relation $< \supseteq E \times E$ such that*

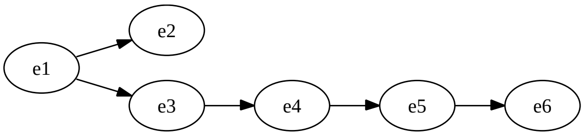
- *$<$ is irreflexive, i.e $\neg(e < e)\ \forall\ e \in E$*
- *$<$ is transitive : $e < e' \wedge e' < e''$ implies $e < e''$*
- *$<$ is acyclic : $e < e' \wedge e' < e$ is forbidden*

Hasse Diagram. *Let $(E, <)$ be a poset. The Hasse diagram $(E, \lessdot)$ is defined by $e \lessdot e'$ iff $e < e'$ and $\neg(\exists e''.\ e < e'' < e')$*

Linearization. *Let $(E, <)$ be a poset. A linearization $(E, <)$ is a total order $\sqsubset$ such that $e < e'$ implies $e \sqsubset e'$.*
A linearization is a topological sort of the Hasse diagram of $(E, <)$.

Example: $E = \{e_1,\ e_2, \ldots, e_n\}$

Hasse diagram:
$< = \{(e_1, e_2), (e_1, e_3), (e_3, e_4), (e_4, e_5), (e_5, e_6), (e_1, e_4), (e_3, e_5), (e_1, e_5), (e_1, e_6), (e_3, e_6), (e_4, e_6)\}$



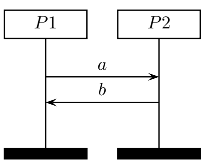
Linearizations: $e_1e_2e_3e_4e_5e_6$, $e_1e_3e_2e_4e_5e_6$, $e_1e_3e_4e_2e_5e_6$, $e_1e_3e_4e_5e_2e_6$, $e_1e_3e_4e_5e_6e_2$, ~~$e_2e_1e_3e_4e_5e_6$~~

2 Message Sequence Graphs

2.1 Message sequence charts

Message sequence charts are a

- Scenario-based language
- Primary use: requirement specification
- Visual Language:



- The notation is standardised by the ITU
- Adopted by the UML (sequence diagrams)
- Widely used in industrial practice
- ‘Easy‘ to comprehend ?

Definition 1 (Message Sequence Chart). *A message sequence chart (MSC) is defined by the tuple M = (P, E, $\mathbb{C}$, l, m, $<$) with :*

- P, *a finite set of processes* $\{P_1,\ P_2, \ldots, P_n\}$
- E, *a finite set of events* $E = \biguplus_{p \in P} E_p = \underbrace{E_? \biguplus E_! \biguplus E_{loc}}_{partitioning\ of\ E}$
- l*:* $E \rightarrow Act$, *a labelling function*

$$l(e) = \begin{cases} p!q(a) & \textit{if } e \in E_p \cap E_!,\ p \neq q,\ a \in \mathbb{C} \\ p?q(a) & \textit{if } e \in E_p \cap E_?,\ p \neq q,\ a \in \mathbb{C} \\ p(a) & \textit{if } e \in E_p \cap E_{loc},\ a \in \mathbb{C} \end{cases}$$

- *m: $E_! \to E_?$, a bijection ("matching function") satisfying :*

$$m(e) = e' \wedge l(e) = p!q(a)\ (p \neq q, a \in \mathbb{C})$$
$$implies \quad l(e') = q?p(a)$$

- *$< \subseteq E \times E$ is a partial order ("visual order")*

$$<= \underbrace{\bigcup_{p \in P} <_p}_{<_p \text{ is a total order "top-to-bottom" order on process p}} \cup \underbrace{\{(e, m(e)) | e \in E_!\}}_{communication\ order\ <_c}$$

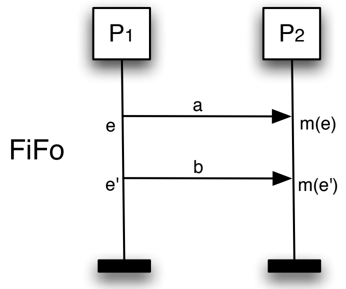
So we have 3 type of events (send, receive, local), a couple of messages. The labelling basically tells you which kind of action happens at each event.
Which send events correspond to which receive events is defined by the matching function $m(e)$. A process is not allowed to send a message to itself. The partial order is an order on the events. It is the combination of the order for all vertical lines (processes) and the horizontal lines (communication).

Definition 2 (FiFo Property)**.** *A MSC M = (P, E, $\mathbb{C}$, l, m, $<$) has the First-Out (FiFo) property whenever for all e, $e' \in E_!$ we have*

$$e < e' \ \wedge\ l(e) = p!q(a)\ \wedge\ l(e') = p!q(b)$$
$$implies \quad m(e) < m(e')$$

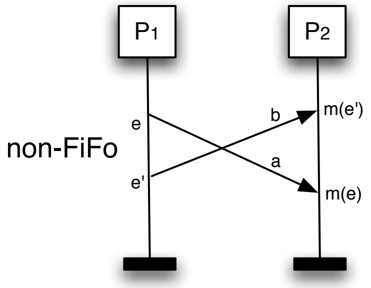
Take two send events (e, e') from P_1 to P_2 and suppose e does happen before e', then the receive event $m(e)$ must happen before the receive event of the second message $m(e')$. A message may not "overtake" another.

Example for an MSC with FiFo property:



with $l(e) = P_1!P_2(a)$
$l(e') = P_1!P_2(b)$
$e < e' \Rightarrow m(e) < m(e')$

Example for an MSC without FiFo property:



Note: We assume an MSC to possess the FiFo property unless stated otherwise.

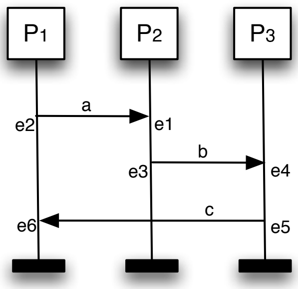
2.1.1 Linearizations

$Lin(M)$ = set of linearizations of MSC M

Lemma 1. *There is a one-to-one correspondence between MSCs and their sets of linearizations.*

In other words, $Lin(M)$ uniquely characteristics M.

2.1.2 Visual Order vs. Possible (Causal) Order



If b takes much shorter than a, then c might arrive at P_1 before a.
Formally:

$$\begin{aligned} <_{P1} &= \{ (e_b, e_2) \} \; is \; possible \\ &\neq visual \; order \end{aligned}$$

So how and when are those situations possible?

2.1.3 Races

Let M = (P, E, $\mathbb{C}$, l, m, $<$) be an MSC.

Definition 3 ($\ll$)**.** *Let $\ll \subseteq E \times E$ be defined by:*

$$
\begin{array}{lll}
e \ll e' & \textit{iff} & e' = m(e) \\
 & \textit{or} & e <_P e' \textit{ and } e \textit{ or } e' \in E_! \qquad\qquad \ll \textit{ is the "interpreted / possible order"} \\
 & \textit{or} & e, e' \in e_P \cap e_? \textit{ and } m^{-1}(e) <_q m^{-1}(e)
\end{array}
$$

We say e is much smaller ($\ll$) than e' if and only if e' is the corresponding receive event to the send event e or the event e happens before e' on the same process P and at least e or e' is a send event
or if we have two receive events on the same process and the corresponding send events happen on another process q with $m^{-1}(e) <_q m^{-1}(e')$.

Example (cf. previous MSC) : $(e_2 \ll e_1, e_3 \ll e_4, e_5 \ll e_6, e_1 \ll e_3, e_4 \ll e_5)$

MSC M contains a <u>race</u> if for some $e, e' \in E_?$

$$e <_P e' \text{ but } \neg(e \ll^* e') \; (\ll^* \text{ is the reflexive and transitive closure of } \ll)$$

We check whether MSC M has a race by computing $\ll^*$ and compare to $<_P$. This is possbile via the Floyd-Warshall Algorithm in $O(|E|^3)$, but as a special case here it is only $O(|E|^2)$.

2.1.4 Computing $\ll^*$ with Warshall's Alorithm

MSC M has a <u>race</u> if $< \not\subseteq \ll^*$ or equivalently:

$$\exists e, e' \in E_? \; (e <_P e' \text{ and } e \not\ll^* e')$$

$\Rightarrow$ protocol implementation based on $<_P$ may cause problems, e.g. unspecified reception, deadlock, or using information from wrong message.

Algorithm: $\underbrace{compute \;\ll^*}_{\text{Warshall's algorithm}}$ and compare with $<$

Warshall's algorithm: <u>input</u> $R \subseteq X \times X$
<u>output</u> R^*

Idea:

consider R and R^* as directed graphs
there is an edge x $\Rightarrow$ y in R^*
iff
there is a (possibly empty) path
$x = x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_n = y$ in R
(our case: $R = \ll$,$R^* = \ll^*$)

2.1.5 Warshalls's Algorithm

Assume the vertices are numbered {1, 2, . . . , n} for $j \in \{1, \ldots n\}$. Define relation $\overset{j}{\Rightarrow}$ as follows:
$x \overset{j}{\Rightarrow} y$ iff $\exists$ path in R from x to y such that all vertices on the path ($\neq x, y$) have a smaller number than j.

Then:

$$x \Rightarrow y \quad \text{iff} \quad x \overset{n+1}{\Rightarrow} y \tag{1}$$

$$x \overset{1}{\Rightarrow} y \quad \text{iff} \quad x = y \tag{2}$$

$$x \overset{y+1}{\Rightarrow} z \quad \text{iff} \quad x \overset{y}{\Rightarrow} z \text{ or } x \overset{y}{\Rightarrow} y \overset{y}{\Rightarrow} z \tag{3}$$

Algorithm: Determine the relations $\overset{1}{\Rightarrow}, \overset{2}{\Rightarrow}, \ldots, \overset{n}{\Rightarrow}, \overset{n,m}{\Rightarrow}$ iteratively using properties $(1) + (2)$.

Store $\overset{i}{\Rightarrow}$ in a boolean matrix C.

Postcondition: $C[x, y] = true$ iff $(x, y) \in R^*$

Precondition: $\forall x,\ y \in X\ C[x, y] = false$

Warshall's Algorithm see Algorithm 1.

Correctness: After j iterations $x \overset{j+1}{\Rightarrow} y$ iff $C[x, y]$.

Proof: by induction on j

Time complexity: $O(n^3)$ where $n = |X|$

2.1.6 Warshalls's Algorithm Efficiency Improvement [Aluret.al '96]

Exploit that for $\ll$:

- $\ll$ is acyclic, and
- the number of predecessors of $e \in E$ under $\ll$ is at most two
  Note: e is an immediate predecessor of e' if:
  $e \ll e'$ and $\neg(\exists e'' \neq e, e'\ e \ll e'' \ll e')$

For body of the algorithm see Algorithm 2.

Algorithm 1 Warshall's Algorithm

for $x := 1$ to n **do** /* Initialization */
  for $y := 1$ to n **do**
    $C[x, y] := (x = y$ or $\underbrace{(x, y) \in R}_{x \ll y})$
    /* loop invariant: */
    /* after the j-th iteration of outermost */
    /* loop it holds $C[x, y]$ iff $x \overset{j+1}{\Rightarrow} y$ */
  od
od
for $y := 1$ to n **do**
  for $x := 1$ to n **do**
    if $C[x, y]$ **then**
      for $z := 1$ to n **do**
        if $C[y, z]$ **then**
          $C[x, z]$ = **true**
        fi
      od
    fi
  od
od

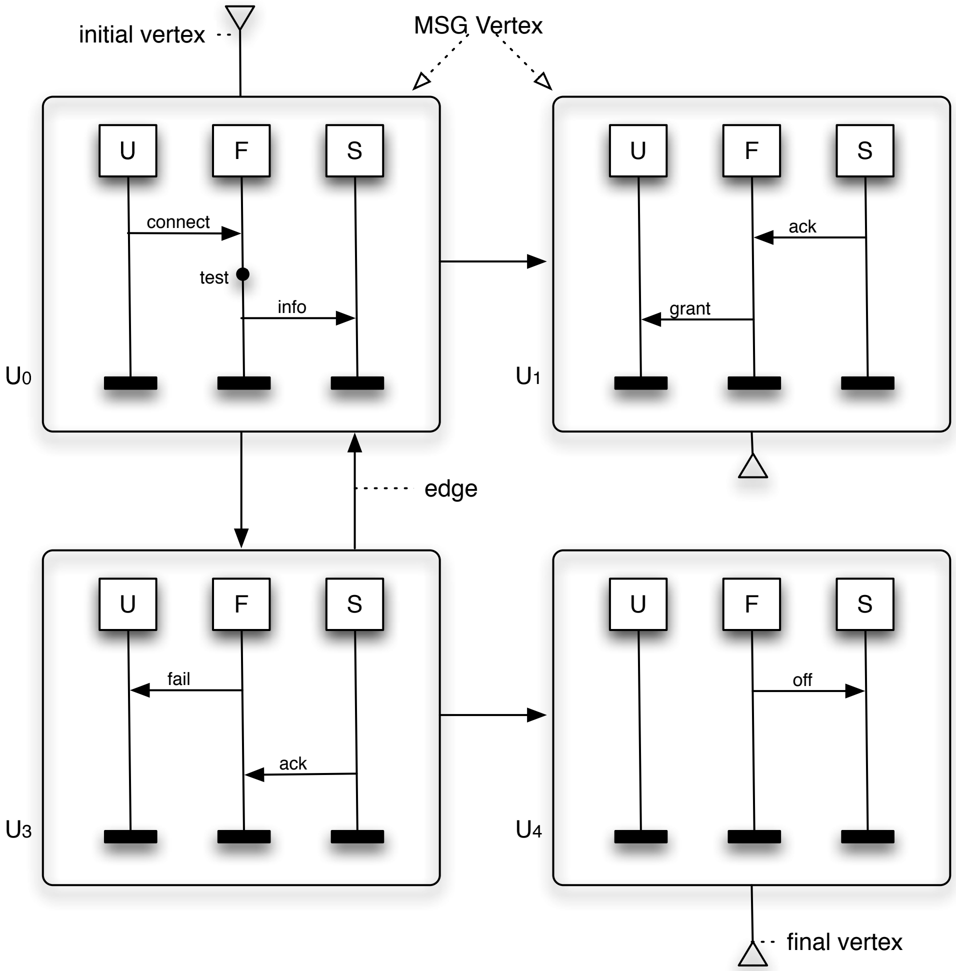
Algorithm 2 Tailored Warshall's Algorithm

for $e := 1$ to n **do**
  for $e' := e - 1$ downto 1 **do**
    if $C[e', e]$ **then**
      /* This part is executed for (e, e') only if e' is an immediate predecessor of e, i.e. $\#innerloops \leq 2 \times n \Rightarrow timecomplexity(O(n^2))$ */
      for $e'' := 1$ to $e' - 1$ **do**
        if $C[e'', e']$ **then**
          $C[e'', e]$ = **true**
        fi
      od
    fi
  od
od

2.2 Sets of MSCs

- MSC specifies a single scenario
- but we typically need a set of scenarios
- and additionally an ordering relation between them, e.g.
  - after scenario 1, scenario 2 occurs
  - after scenario 1, scenario 2 or 3 occurs
  - scenario 1 occurs repeatingly
- So we need :
  - alternative composition,
  - sequential composition (=concatenation),
  - iteration of MSCs

  This yields **Message Sequence Graphs**
- Alternatives are ensembles of MSCs and high-level MSCs (MSC'96)



$$U_0 \cdot U_2 \cdot U_0 \cdot U_1 = \lambda(U_0) \cdot \lambda(U_2) \cdot \lambda(U_0) \cdot \lambda(U_1)$$

Definition 4 (Message Sequence Graph)**.** *Let $\mathbb{M}$ be the set of MSCs. (up to isomorphism i.e. event identities). A Message Sequence Graph (MSG) G is a tuple $G = (V, \rightarrow, v_o, F, \lambda)$ with:*

- *$(V, \rightarrow)$ is a digraph with a finite set V of vertices and $\rightarrow \subseteq V \times V$ the set of edges.*
- *$v_0 \in V$ is the starting (or initial) vertex*
- *$F \subseteq V$ is a set of final vertices*
- *$\lambda : V \longrightarrow \mathbb{M}$ associates to each vertex $v \in V$ an MSC $\lambda(v)$*

Note:

- An MSG is an NFA without input alphabet where states are MSCs
- Every MSC is an MSG

2.2.1 Concatenation of MSCs

Let M_i = (P_i, E_i, $\mathbb{C}_i$, l_i, m_i, $<_i$) with $i \in \{1,\ 2\}$ be an MSC with $E_1 \cap E_2 = 0$.

The (weak) concatenation of M_1 and M_2 is the MSC $M_1 \cdot M_2$ = (P, E, $\mathbb{C}$, l, m, $<$) with

$P = P_1 \cup P_2 \quad E = E_1 \cup E_2 \quad \mathbb{C} = \mathbb{C}_1 \cup \mathbb{C}_2$
with $E_? = E_{1?} \cup E_{2?}$, $E_! = E_{1!} \cup E_{2!}$, $E_{loc} = E_{loc?} \cup E_{loc?}$. And

$$l(e) = \begin{cases} l_1(e) & \text{if } e \in E_1 \\ l_2(e) & \text{if } e \in E_2 \end{cases}$$

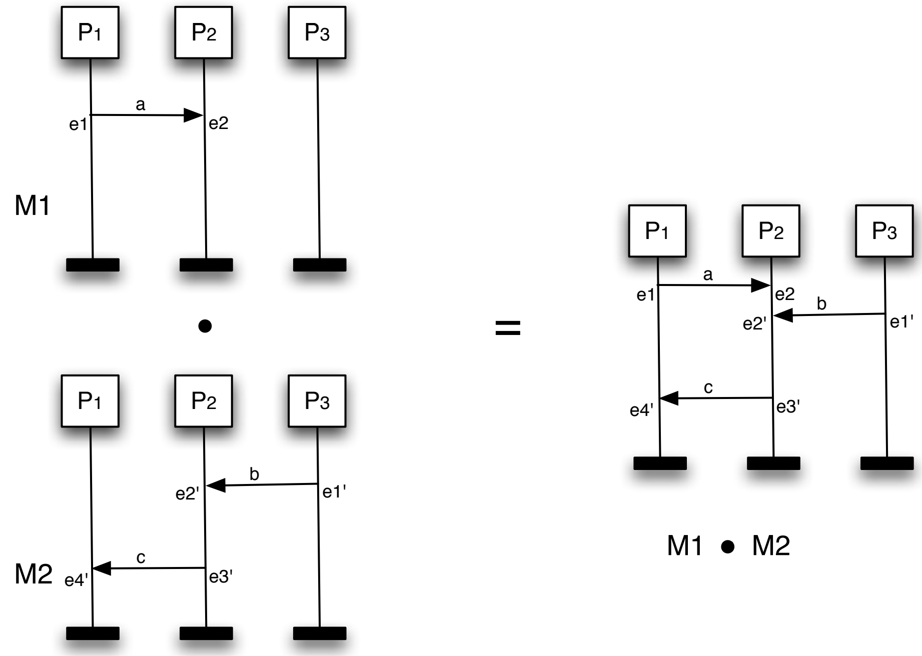
$$m(e) = \begin{cases} m_1(e) & \text{if } e \in E_1 \\ m_2(e) & \text{if } e \in E_2 \end{cases}$$

$<=<_1 \cup <_2 \cup \{(e, e')|E_1 \cap E_P,\ e' \in E_2 \cap E_P\}$ where P is the same process both times.

Note:

- Events are ordered process-wise: Events at p in M_1 precede events at p in M_2
- Thus: some processes may proceed to M_2 before others.
- $\neq$: first complete M_1 then execute M_2. (Strong concatenation is covered in Assignment 1, Excercise 4.)

Some examples of concatenation :



Note: e_1 and e_1' are not ordered in $M_1 \cdot M_2$, e.g.

$$e_1, e_2, e_1', e_2', \dots \in Lin(M_1 \cdot M_2)$$
$$e_1', e_1, e_2, e_2', \dots \in Lin(M_1 \cdot M_2)$$

2.2.2 Language of an MSG

Let $G = (V, \rightarrow, v_o, F, \lambda)$ be an MSG.
A path π of G is a finite sequence

$$\pi = u_0, u_1, \dots, u_n \text{ with } u_i \in V, 0 \leq i \leq n \text{ and } u_i \rightarrow u_{i+1}, 0 \leq i \leq n$$

The MSC of a path π = $u_0, u_1, \ldots, u_n$ is:

$$M(\pi) = \lambda(u_0) \cdot \lambda(u_1) \cdot \ldots \cdot \cdot \lambda(u_n) \quad \text{, where } \cdot \text{ is the MSC concatenation operator}$$
$$= \prod_{i=0}^{n} \lambda(u_i)$$

The path $\pi = u_0, u_1, \ldots, u_n$ is accepting if $u_0 = v_0$ and $u_n \in F$

The (MSC) language of MSG G is defined by :
$L(G) = \{M(\pi) \mid \pi$ is an accepting path of $G\}$

The word language of MSG G is $Lin(L(G))$ where:
$Lin(\{M_1, \ldots, M_k\}) = \bigcup_{i=1}^{k} Lin(M_i)$

2.2.3 Races in MSGs

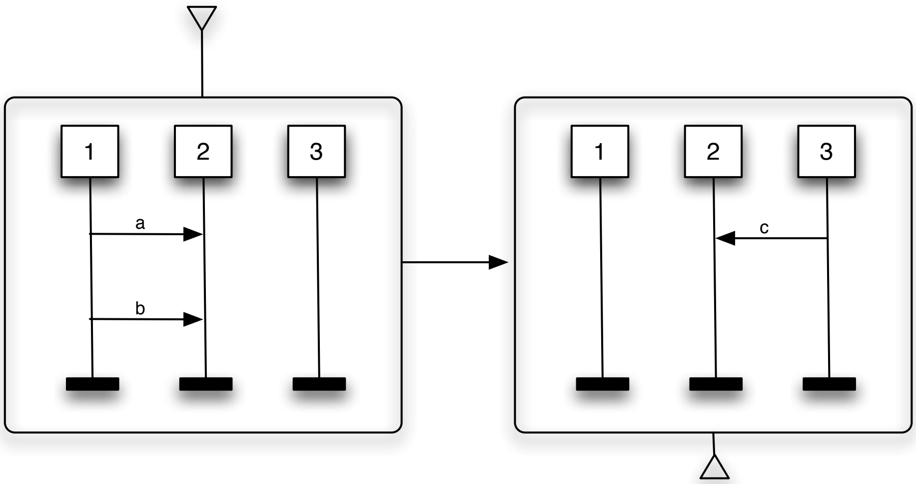
Recall: MSC M has a race if $< \not\subseteq \ll^*$
or equivalently $Lin(E, <) \not\subseteq Lin(E, \ll^*)$
or equivalently $Lin(E, <) \subset Lin(E, \ll^*)$
(since $Lin(E, <) \subseteq Lin(E, \ll^*)$)

Definition 5 (Races in MSGs)**.** *MSG G has a race if* $Lin(G, <) \subseteq Lin(G, \ll^*)$ *[Musholl, Peled '99]*

Theorem 1 (without proof)**.** *The decision problem "MSG G has a race" is undecidable.*

Proof. Reduction from Post's correspondence problem (PCP). This reduction is not easy, however we will see a similar - though simpler - proof later on for a different problem. □

Example of an MSG that has a race:



Each individua; MSC is race-free, but their concatenation is not.

3 Expressiveness of MSGs

Three facts about the expressiveness of MSGs.

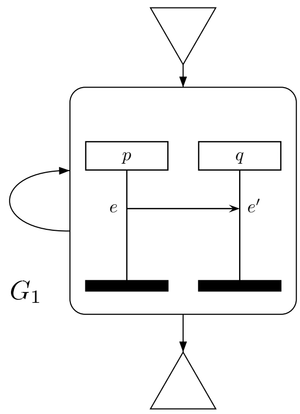
3.1 MSGs may represent infinite-state systems

Definition 6 (States of MSCs)**.** *The* state *of an MSC with event set E is $E' \subseteq E$, such that $e \in E' \wedge e' < eimpliese' \in E'$. In other words, E' is downward-closed wrt. $<$.*

Definition 7 (State space of MSCs)**.** *The set of states of an MSC M is the* state space *of M.*

Definition 8 (State space of MCGs)**.** *The* state space *of an MSG G is the union of the state spaces of M_i with $M_i \in L(G)$*

3.1.1 Example

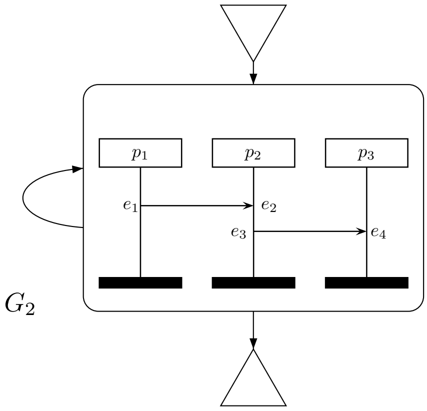


G_1 has an infinite state space.

A possible state is $\{e^{(1)}, e^{(2)}, e^{(3)}, \ldots\}$ with $e^{(i)}$ the occurrence of e in the i-th iteration.

⇒ A system that realizes G would require an *unbounded* communication channel.

3.2 State space of MSG may not be context-free



The states of G_2 are of the form $\{e_1^k, e_2^l, e_3^m, e_4^n | k \geq l \geq m \geq n\}$.
The corresponding language is not context-free.

3.3 The state space of an MSG is context-sensitive

Let M be an MSC with event set E with $e, e' \in E$ and $w, w' \in E^*$. Consider the following rules:

(1) $wee'w' \in Lin(M)$, $l(e) = q?p(b)$, $l(e') = p!q(a)$ implies $we'ew' \in Lin(M)$. Note that the reverse does not always hold.

(2) $wee'w' \in Lin(M)$, $l(e) = p!q(a)$, $l(e') = q?p(b)$ and

$$\underbrace{\sum_{a \in \mathbb{C}} |w|_{p!q(a)}}_{\text{number of sends from p to q in w}} > \underbrace{\sum_{a \in \mathbb{C}} |w|_{q?p(a)}}_{\text{number of receipts of q from p in w}}$$

implies $we'ew' \in Lin(M)$.

(3) $wee'w' \in Lin(M)$, $e \in E_p$, $e' \in E_q$, $p \neq q$ and e, e' do not match like in (1) + (2) implies $we'ew' \in Lin(M)$

Note: Rule (2) is a *context-sensitive* rule of form $X\underline{ab}Y \rightarrow X\underline{ba}Y$

3.4 Context sensitivity (informal argument)

- Take MSG G and use vertex identities as vertex labels.
- $K(G) =$ set of ”accepting” vertex seqeuences.

- Replace each vertex v by $Lin(\lambda(v))$ (interpret sequencing element wise)
- Let the resulting set be $\tilde{K}(G)$
- Close $\tilde{K}(G)$ under the permutation rules (1), (2) and (3) as described in the previous section.

The resulting language is context sensitive.

4 Intersection of MSGs

Theorem 2 (The emptiness problem of the intersection of MSGs is undecidable). *Let G_1 and G_2 be MSGs. The decision problem $L(G_1) \cap L(G_2) = \emptyset$ is undecidable.*

Proof

Reduction from Post's Correspondence Problem.

Definition 9 (Post's Correspondence Problem (PCP)).
Input: $\{(u_1, w_1), (u_2, w_2), \ldots, (u_n, w_n)\}$ *with* $u_i, w_i \in \Sigma^*$ *for some alphabet* Σ, $1 <= i <= n$
Decision problem: Does there exist a sequence of indexes $i_1, \ldots, i_k$ *with* $1 \leq i_j \leq n$ *with* $1 <= j <= k$*, such that* $u_{i_1} u_{i_2} \ldots u_{i_n} = w_{i_1} w_{i_2} \ldots w_{i_n}$

Example

Input: $\{(\underbrace{aba}_{u_1}, \underbrace{a}_{w_1}), (\underbrace{bbb}_{u_2}, \underbrace{aaa}_{w_2}), (aab, abab), (bb, babba)\}$

One solution would be the index sequence $1, 4, 3, 1$ with $\underbrace{aba}_{u_1}\underbrace{bb}_{u_4}\underbrace{aab}_{u_3}\underbrace{aba}_{u_1} = \underbrace{a}_{w_1}\underbrace{babba}_{w_4}\underbrace{abab}_{w_3}\underbrace{a}_{w_1}$

Theorem 3 (Undecidability of PCP). *Post's Correspondence Problem is undecidable.*

Definition 10 (Reduction technique). *Let P, Q be decision problems. If P is reducible to Q and P is undecidable, then Q is undecidable.*

In our case, let $P \hat{=} PCP$ and Q the intersection problem.

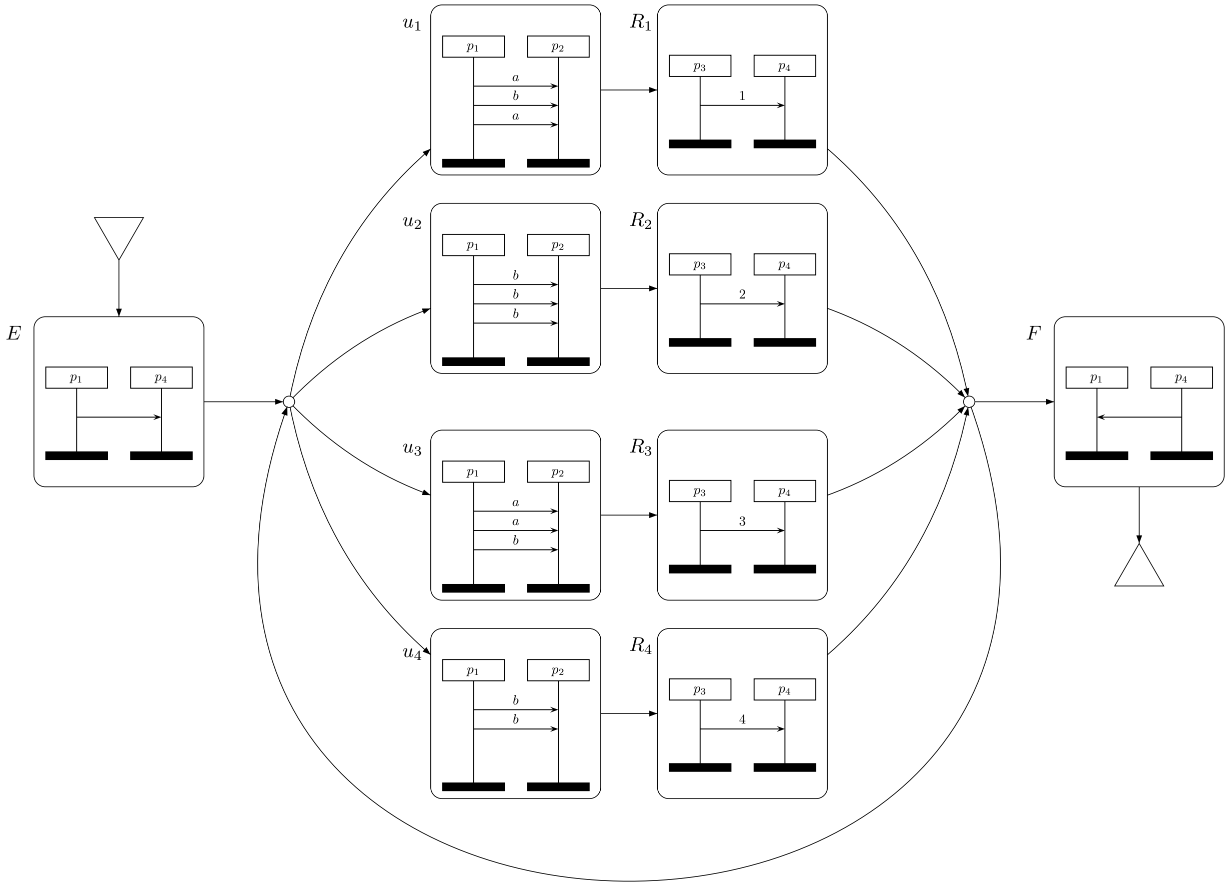
Find a transformation from an instance $\{(u_1, w_1), \ldots, (u_n, w_n)\}$ of PCP to MSGs G_u, G_w, such that PCP has a solution iff $L(G_u) \cap L(G_w) \neq \emptyset$.

Instead of a formal definition, the construction of the MSGs will be explained by an example:

For the PCP instance, let

$u_1 = aba$
$u_2 = bbb$
$u_3 = aab$
$u_4 = bb$

Construct G_u with processes $P = \{P_1, \ldots P_4\}$



The corresponding language is $L(G_u) = E \cdot (\sum_{j=1}^{n}(M_j \cdot R_j))^+ \cdot F$. Infact $\lambda(E) \cdot (\sum_{j=1}^{n} \lambda(u_j) \cdot \lambda(R_j))^+ \cdot \lambda(F)$. G_w is constructed in a similar fashion. It can now be shown that :

PCP on $\{(u, w), \ldots, (u_n, w_n)\}$ has a solution iff $L(G_u) \cap L(G_w) \neq 0$.