Overview

automata

1. Lecture 3 + 4 : $\boxed{\text{Message Sequence Graphs}}$

Theoretical Foundations of the UML

Lecture 3+4: Message Sequence Graphs

Joost-Pieter Katoen

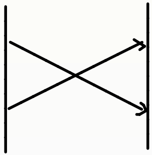
Lehrstuhl für Informatik 2
Software Modeling and Verification Group

moves.rwth-aachen.de/teaching/ss-20/fuml/

April 28, 2020

Summary of Lecture #3

1. A Message Sequence Chart is a visual partial order
    - between send and receive events
    - totally ordered per process — vertical ordering
    - receive events happen after their send events — horizontal ordering
    - respecting the FIFO property

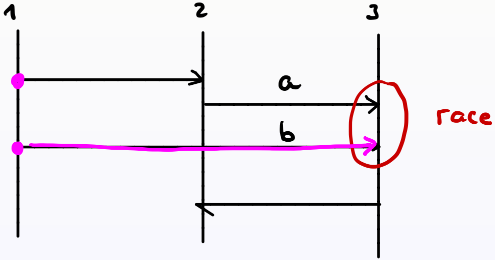


non-FIFO

Summary of Lecture #3

1. A Message Sequence Chart is a visual partial order
   - between send and receive events
   - totally ordered per process — vertical ordering
   - receive events happen after their send events — horizontal ordering
   - respecting the FIFO property

2. Race: in practice, $\underbrace{\text{the order of receive events}}_{\text{as indicated by the MSC}}$ cannot be guaranteed

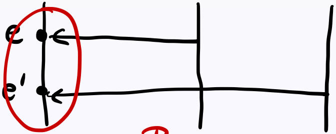


Summary of Lecture #3

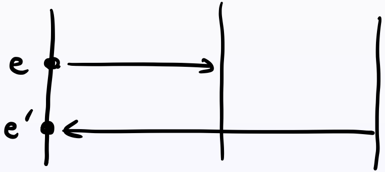
1. A Message Sequence Chart is a visual partial order $\leq^*$
   - between send and receive events
   - totally ordered per process — vertical ordering
   - receive events happen after their send events — horizontal ordering
   - respecting the FIFO property

2. Race: in practice, the order of receive events cannot be guaranteed

3. Causal order $\ll^*$
   - send events should happen before their matching receive events ✓
   - the ordering of events wrt. sends on same process is respected
   - receive events on a process sent from the same process are ordered as their sends



B



A

Summary of Lecture #3

1. A Message Sequence Chart is a visual partial order
   - between send and receive events
   - totally ordered per process — vertical ordering
   - receive events happen after their send events — horizontal ordering
   - respecting the FIFO property
2. Race: in practice, the order of receive events cannot be guaranteed
3. Causal order
   - send events should happen before their matching receive events
   - the ordering of events wrt. sends on same process is respected
   - receive events on a process sent from the same process are ordered as their sends
4. A MSC has a race if causal order $\neq$ visual order

Summary of Lecture #3

1. A Message Sequence Chart is a visual partial order
   - between send and receive events
   - totally ordered per process — vertical ordering
   - receive events happen after their send events — horizontal ordering
   - respecting the FIFO property

2. Race: in practice, the order of receive events cannot be guaranteed

3. Causal order
   - send events should happen before their matching receive events
   - the ordering of events wrt. sends on same process is respected
   - receive events on a process sent from the same process are ordered as their sends

4. A MSC has a race if causal order $\neq$ visual order
   - checking whether an MSC has a race can be done in quadratic time (in number of events)
   - using an optimized version of Warshall's algorithm

$O(|E|^3)$

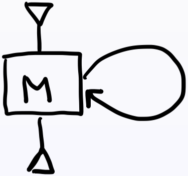
The need for composing MSCs

- An MSC describes a possible single scenario
- Typically: a set of scenarios

$\underbrace{\{M_1, \ldots, M_k\}}_{\text{finite}}$ $k \in \mathbb{N}$

infinite

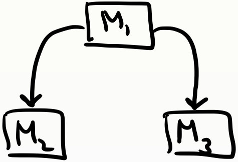
$\underbrace{M \quad \text{scenario}}_{\text{arbitrarily many times}}$ M^*



MS graph

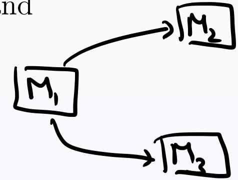
The need for composing MSCs

- An MSC describes a possible single scenario
- Typically: a set of scenarios
- and dependencies between these scenarios:
  - after scenario 1, scenario 2 occurs
  - after scenario 1, scenario 2 or 3 occurs
  - scenario 1 occurs repeatedly



The need for composing MSCs

- An MSC describes a possible single scenario
- Typically: a set of scenarios
- and dependencies between these scenarios:
  - after scenario 1, scenario 2 occurs
  - after scenario 1, scenario 2 or 3 occurs
  - scenario 1 occurs repeatedly
- Need for: sequential composition (= concatenation), alternative composition, and iteration of MSCs



The need for composing MSCs

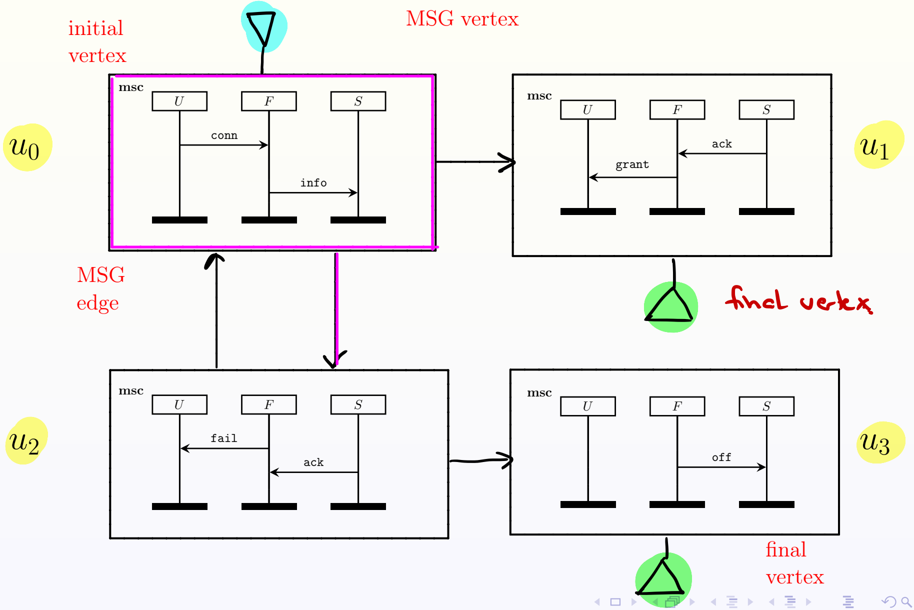
- An MSC describes a possible single scenario
- Typically: a set of scenarios
- and dependencies between these scenarios:
  - after scenario 1, scenario 2 occurs
  - after scenario 1, scenario 2 or 3 occurs
  - scenario 1 occurs repeatedly
- Need for: sequential composition (= concatenation), alternative composition, and iteration of MSCs

⇒ This yields Message Sequence Graphs

aka: hierarchical MSCs or message sequence chart automata

- Alternatives: ensembles of MSCs, high-level MSCs (MSC'96)

Message Sequence Graphs



Message Sequence Graphs

Let $\mathbb{M}$ be the set of MSCs (up to isomorphism, i.e., event identities).

Definition

A Message Sequence Graph (MSG) $G = (V, \rightarrow, v_0, F, \lambda)$ with:

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

Message Sequence Graphs

Let $\mathbb{M}$ be the set of MSCs (up to isomorphism, i.e., event identities).

Definition

A Message Sequence Graph (MSG) $G = (V, \rightarrow, v_0, F, \lambda)$ with:

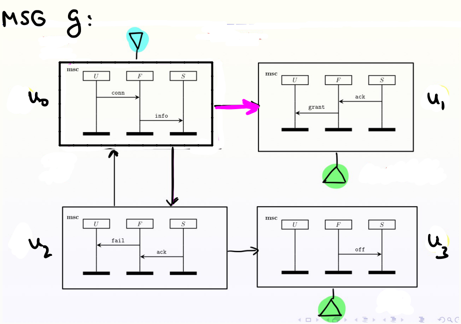
- $(V, \rightarrow)$ is a digraph with finite set V of vertices and $\rightarrow \subseteq V \times V$ a set of edges
- $v_0 \in V$ is the starting (or: initial) vertex
- $F \subseteq V$ is a set of final vertices
- $\lambda : V \rightarrow \mathbb{M}$ associates to each vertex $v \in V$, an MSC $\lambda(v)$

Note:

An MSG can be considered as a non-deterministic finite-state automaton without input alphabet where states are MSCs. Obviously, every MSC is an MSG.

Example

MSG $\mathcal{G}$:



$$\mathcal{G} = (V, \rightarrow, v_0, F, \lambda)$$

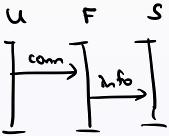
$$V = \{u_0, \ldots, u_3\}$$

$$\rightarrow = \{(u_0,u_1), (u_0,u_2), (u_2,u_0), (u_2,u_3)\}$$

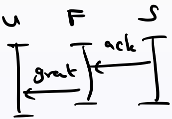
$$v_0 = u_0$$

$$F = \{u_1, u_3\}$$

$\lambda(u_0) = M_0 =$

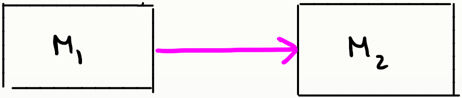


$\lambda(u_1) =$



Concatenation of MSCs: definition

Let $M_i = (\mathcal{P}_i, E_i, \mathcal{C}_i, l_i, m_i, \preceq_i)$ with $i \in \{1, 2\}$
be two MSCs with $E_1 \cap E_2 = \varnothing$



Concatenation of MSCs: definition

Let $M_i = (\mathcal{P}_i, E_i, \mathcal{C}_i, l_i, m_i, \preceq_i)$ with $i \in \{1, 2\}$
be two MSCs with $E_1 \cap E_2 = \varnothing$

The concatenation of M_1 and M_2 is the MSC
$M_1 \bullet M_2 = (\mathcal{P}, E, \mathcal{C}, l, m, \preceq)$ with:

$$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \qquad E = E_1 \cup E_2 \qquad \mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$$

(with $E_? = E_{1,?} \cup E_{2,?}$ etc.)

$$l(e) = \begin{cases} l_1(e) & \text{if} \quad e \in E_1 \\ l_2(e) & \text{if} \quad e \in E_2 \end{cases} \qquad m(e) = \begin{cases} m_1(e) & \text{if} \quad e \in E_1 \\ m_2(e) & \text{if} \quad e \in E_2 \end{cases}$$

$e \in E!$

Concatenation of MSCs: definition

Let $M_i = (\mathcal{P}_i, E_i, \mathcal{C}_i, l_i, m_i, \preceq_i)$ with $i \in \{1, 2\}$
be two MSCs with $E_1 \cap E_2 = \varnothing$

The concatenation of M_1 and M_2 is the MSC
$M_1 \bullet M_2 = (\mathcal{P}, E, \mathcal{C}, l, m, \preceq)$ with:

$$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \qquad E = E_1 \cup E_2 \qquad \mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$$

(with $E_? = E_{1,?} \cup E_{2,?}$ etc.)

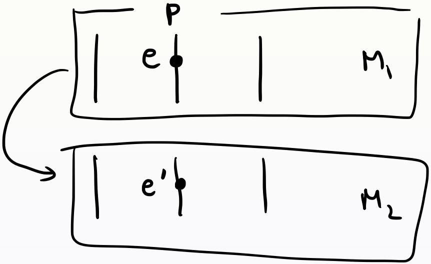
$$l(e) = \begin{cases} l_1(e) & \text{if} \quad e \in E_1 \\ l_2(e) & \text{if} \quad e \in E_2 \end{cases} \qquad m(e) = \begin{cases} m_1(e) & \text{if} \quad e \in E_1 \\ m_2(e) & \text{if} \quad e \in E_2 \end{cases}$$

$$\preceq \; = \Big(\preceq_1 \cup \preceq_2 \cup \underbrace{\{(e, e') \mid \exists p \in \mathcal{P}. \, e \in E_1 \cap E_p, \, e' \in E_2 \cap E_p\}}_{\text{new ordering}}\Big)^*$$

Ordering

$$\preceq \ = \left(\preceq_1 \cup \preceq_2 \cup \underbrace{\{(e,e') \mid \exists p \in \mathcal{P}.\, e \in E_1 \cap E_p \,,\ e' \in E_2 \cap E_p\}}\right)^*$$

$M_1 \bullet M_2$ $\quad M_1 \quad M_2$



Concatenation of MSCs: observations

Ordering

$\preceq \;=\; \left(\preceq_1 \cup \preceq_2 \cup \{(e, e') \mid \exists p \in \mathcal{P}.\, e \in E_1 \cap E_p\,,\; e' \in E_2 \cap E_p\}\right)^*$

Observations

- events are ordered per process:

  every event at $\underline{p}$ in MSC $\underline{M_1}$ precedes every event at $\underline{p}$ in MSC $\underline{M_2}$

Concatenation of MSCs: observations

Ordering

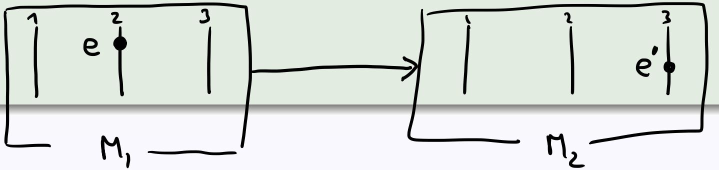
$$\preceq \;=\; \left(\preceq_1 \cup \preceq_2 \cup \{(e,e') \mid \exists p \in \mathcal{P}.\, e \in E_1 \cap E_p,\ e' \in E_2 \cap E_p\}\right)^*$$

Observations

- events are ordered per process:

  every event at p in MSC M_1 precedes every event at p in MSC M_2

- events at distinct processes in M_1 and M_2 can be incomparable wrt $\preceq$



Concatenation of MSCs: observations

Ordering

$$\preceq \; = \Big(\preceq_1 \cup \preceq_2 \cup \{ (e, e') \mid \exists p \in \mathcal{P}.\, e \in E_1 \cap E_p \,,\; e' \in E_2 \cap E_p \} \Big)^*$$

Observations

- events are ordered per process:

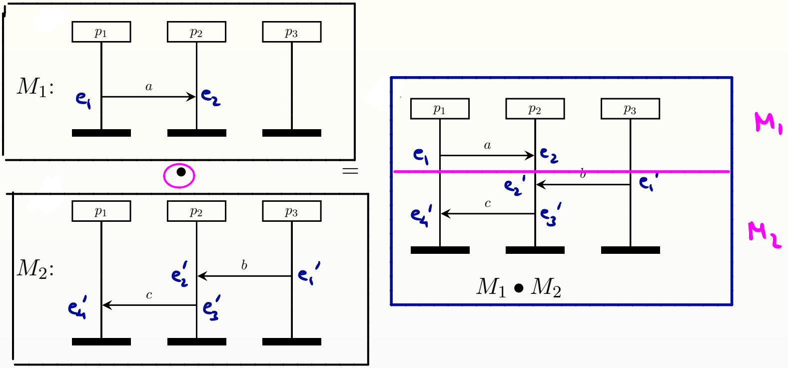
  every event at p in MSC M_1 precedes every event at p in MSC M_2

- events at distinct processes in M_1 and M_2 can be incomparable
- thus: a process may start with M_2 before other processes do pause
- this differs from: first complete M_1, then start with M_2

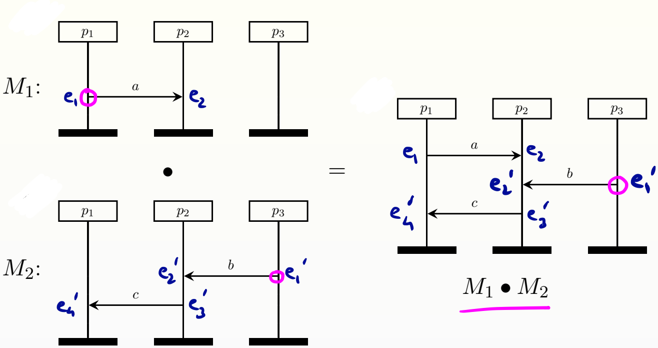
have finished M_1

$\neq$ execute all events in M_1

Example (1)



Example (1)



$M_1 \bullet M_2$

$$\preceq_1 : \begin{tikzcd} e_1 \arrow[r] & e_2 \end{tikzcd}$$

$$\begin{tikzcd} e_2' \arrow[d] & e_1' \arrow[l] \\ e_3' & \end{tikzcd}$$

$$\preceq_2 : \begin{tikzcd} e_4' & e_3' \arrow[l] \end{tikzcd}$$

$$\preceq : \begin{tikzcd} e_1 \arrow[r] \arrow[dd] & e_2 \arrow[d] & \\ & e_2' \arrow[d] & e_1' \arrow[l] \\ e_4' & e_3' \arrow[l] & \end{tikzcd}$$

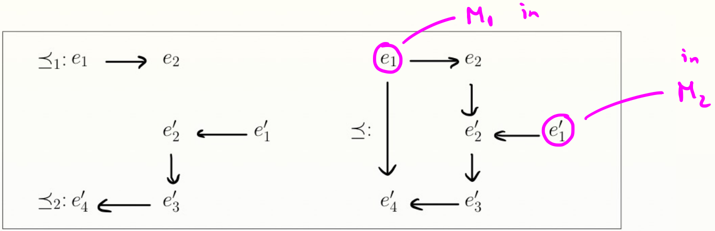
$M_1 \bullet M_2$

Example (2)

$$
\begin{tikzcd}
\preceq_1\colon e_1 \arrow[r] & e_2 & \\
 & e'_2 \arrow[d] & e'_1 \arrow[l] \\
\preceq_2\colon e'_4 & e'_3 \arrow[l] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & e_1 \arrow[r] \arrow[dd] & e_2 \arrow[d] & \\
\preceq\colon & & e'_2 \arrow[d] & e'_1 \arrow[l] \\
 & e'_4 & e'_3 \arrow[l] &
\end{tikzcd}
$$

$M_1 \cdot M_2$

Example (2)



Note:

Events e_1 and e'_1 are not ordered in $M_1 \bullet M_2$

Example linearizations:

$e_1 \quad e_2 \quad e'_1 \quad e'_2 \quad \ldots \in Lin(M_1 \bullet M_2)$

$e'_1 \quad e_1 \quad e_2 \quad e'_2 \quad \ldots \in Lin(M_1 \bullet M_2)$

Properties of concatenation

1. Concatenation is associative:

$$\underbrace{\underbrace{(M_1 \bullet M_2)} \bullet M_3} \; = \; \underbrace{M_1 \bullet \underbrace{(M_2 \bullet M_3)}}$$

$M_1 \bullet M_2 \bullet \ldots \bullet M_k$

Properties of concatenation

1. Concatenation is associative:

$$(M_1 \bullet M_2) \bullet M_3 \;=\; M_1 \bullet (M_2 \bullet M_3)$$

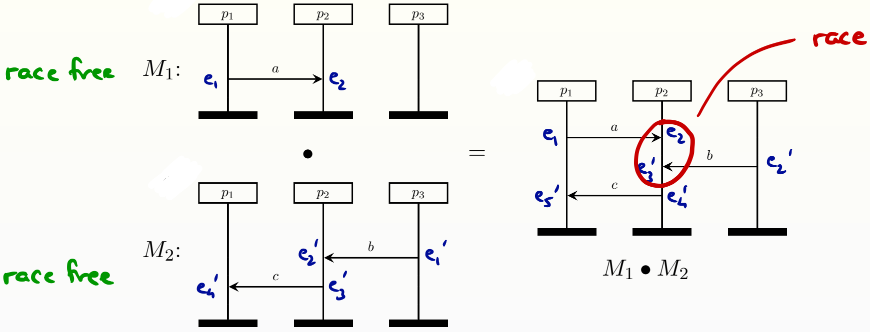
2. Concatenation preserves the FIFO property:

$$(M_1 \text{ is FIFO } \wedge M_2 \text{ is FIFO }) \quad \text{implies} \quad M_1 \bullet M_2 \text{ is FIFO}$$

3. Race-freeness, however, is not preserved

$$(\underbrace{M_1 \text{ is race-free}} \wedge \underbrace{M_2 \text{ is race-free}}) \quad \not\Rightarrow \quad \underbrace{M_1 \bullet M_2} \text{ is race-free}$$

Example Race Freedom



$$\preceq_1 : e_1 \longrightarrow e_2$$

$$\begin{tikzcd}
 & e_2' \arrow[d] & e_1' \arrow[l] \\
\preceq_2 : e_4' & e_3' \arrow[l] &
\end{tikzcd}$$

$$\preceq : \begin{tikzcd}
e_1 \arrow[r] \arrow[dd] & e_2 \arrow[d] & \\
 & e_2' \arrow[d] & e_1' \arrow[l] \\
e_4' & e_3' \arrow[l] &
\end{tikzcd}$$

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

A path through MSG G is a finite traversal through the graph G.

Definition

A path π in MSG G is a finite sequence

$$\pi = \underbrace{u_0 \; u_1 \; \ldots \; u_n}_{} \text{ with } \underbrace{u_i \in V}_{} \quad (0 \leq i \leq n) \text{ and } \underbrace{u_i \rightarrow u_{i+1}}_{} \quad (0 \leq i < n)$$

$(u_i, u_{i+1}) \in \rightarrow$

Paths in MSGs

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

A path through MSG G is a finite traversal through the graph G.

Definition

A path π in MSG G is a finite sequence

$$\pi = u_0 \; u_1 \; \ldots \; u_n \text{ with } u_i \in V \;\; (0 \leq i \leq n) \text{ and } u_i \rightarrow u_{i+1} \;\; (0 \leq i < n)$$

An accepting path through MSG G is a path starting in the initial vertex and ending in a final vertex.

Definition

Path $\pi = u_0 \; \ldots \; u_n$ is accepting if: $u_0 = v_0$ and $u_n \in F$.

$$\boxed{u_0} \to \boxed{u_1} \to \boxed{u_2} \;- - - - - \to \boxed{u_{n-1}} \to \boxed{u_n}$$

$= v_0$ $\qquad\qquad\qquad\qquad$ $\in F$

Let $G = (V, \to, v_0, F, \lambda)$ be an MSG.

λ

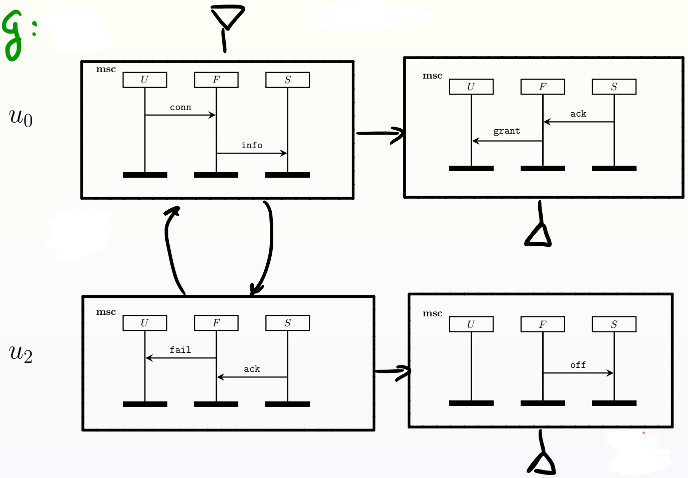
Definition

The MSC of a path $\pi = u_0 \; \ldots \; u_n$ through MSG G is defined by:

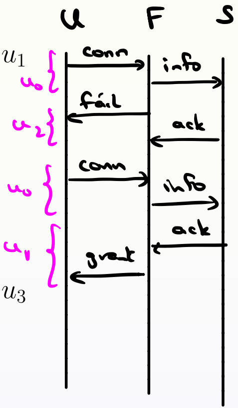
$$M(\pi) \;=\; \underbrace{\lambda(u_0)}_{\text{MSC of } u_0} \;\bullet\; \underbrace{\lambda(u_1)}_{\text{MSC of } u_1} \;\bullet \ldots \bullet\; \underbrace{\lambda(u_n)}_{\text{MSC of } u_n}$$

Example paths

$\mathcal{G}$:



$M(u_0\, u_2\, u_0\, u_1)$



Accepting: $u_0\, u_1$

$u_0\, u_2\, u_0\, u_1$

$u_0\, u_2\, u_0\, u_2\, u_3$

Non-accepting: $u_2\, u_3$

$u_0\, u_2\, u_0\, u_2$

etc.

Language of an MSG

- set of MSCs that are accepted by the MSG

- MSG $\mathcal{G}$ as a descriptor a set of MSCs

finite

infinite

Language of an MSG

The language of an MSG, i.e., the set of MSCs it represents, is the set of MSCs of its accepting paths.

Definition

The MSC language of MSG G is defined by:

$$L(G) = \{\underbrace{M(\pi) \mid \pi \text{ is an accepting path of } G}_{\text{the MSCs of accepting paths in } G}\}.$$

Language of an MSG

The language of an MSG, i.e., the set of MSCs it represents, is the set of MSCs of its accepting paths.

Definition

The MSC language of MSG G is defined by:

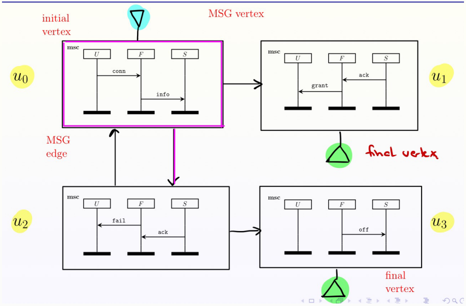
$$L(G) = \{ M(\pi) \mid \pi \text{ is an accepting path of } G \}.$$

Definition

The word language of MSG G is defined by $Lin(L(G))$ where

$$Lin(\underbrace{\{M_1, \ldots, M_k\}}_{L(G)}) = \bigcup_{i=1}^{k} Lin(M_i).$$

Example



accepted paths:

$u_0 (u_2 u_0)^* u_1$,

$u_0 (u_2 u_0)^* u_2 u_3$

where $*$ means arbitrarily many (but finitely many)

MSC $\mathcal{G}$

$$L(\mathcal{G}) = \{ M(u_0 (u_2 u_0)^* u_1), \; M(u_0 (u_2 u_0)^* u_2 u_3) \}$$

$$|L(G)| = \infty$$

Races in MSGs

Recall: MSC M has a race if $\preceq \not\subseteq \ll^*$

visual order

causal order

Races in MSGs

Recall: MSC M has a race if $\preceq \not\subseteq \ll^*$

or, equivalently $\quad \underbrace{Lin(M, \preceq)}_{\text{linearisations wrt } \preceq} \not\subseteq \underbrace{Lin(M, \ll^*)}_{\text{linearisations wrt } \ll^*}$

$\neq$

Races in MSGs

Recall: MSC M has a race if $\preceq \not\subseteq \ll^*$

or, equivalently $\quad Lin(M, \preceq) \not\subseteq Lin(M, \ll^*)$

or, equivalently $\quad Lin(M, \ll^*) \subset Lin(M, \preceq)$

Definition

MSG G has a race if $Lin(G, \ll^*) \subset Lin(G, \preceq)$

$\neq$

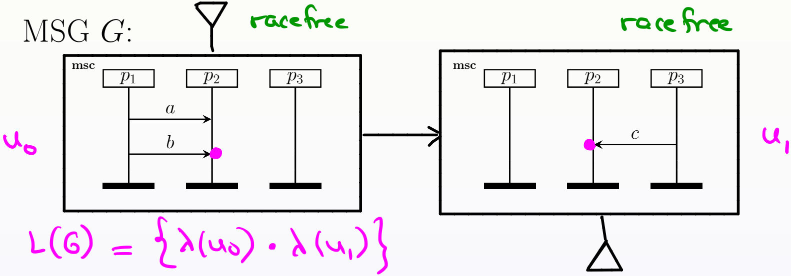
MSG $\mathcal{G}$ has race if some MSC $M \in L(G)$ has a race.

Example

Definition

MSG G has a race if $Lin(G, \ll^*) \subset Lin(G, \preceq)$

MSG G:



$L(G) = \{\lambda(u_0) \cdot \lambda(u_1)\}$

MSG G has a race.

Deciding whether an MSG has a race is undecidable

Does MSC M have a race? $\longrightarrow$ Worshall algorithm

$\mathcal{O}(|E|^2)$

events in M

Does MSG $\mathcal{G}$ have a race?

– if $L(\mathcal{G}) = \{M_1, \ldots, M_k\}$ $k \in \mathbb{N}$.

run Worshall's on each M_i

$\mathcal{O}(k \cdot |E|^2)$

– if $L(\mathcal{G}) = \{M_1, \ldots\}$

Deciding whether an MSG has a race is undecidable

Theorem [Muscholl & Peled, 1999]

The decision problem "does MSG G have a race?" is undecidable.

Proof.

By a reduction from the universality of semi-trace languages. Requires some Mazurkiewicz' trace theory. Omitted here. We will see other reduction proofs later on. □

$L = \Sigma^* \,?$ universality problem

Deciding whether an MSG has a race is undecidable

Theorem [Muscholl & Peled, 1999]

The decision problem "does MSG G have a race?" is undecidable.

Proof.

By a reduction from the universality of semi-trace languages. Requires some Mazurkiewicz' trace theory. Omitted here. We will see other reduction proofs later on. $\square$

No undecidable problem can ever be solved by a computer or computer program of any kind.

Do MSGs have an MSC in common?

Theorem: undecidability of empty intersection

The decision problem:

$$\text{for MSGs } \underline{G_1} \text{ and } \underline{G_2}\text{, do we have } \underline{L(G_1)} \cap \underline{L(G_2)} = \varnothing?$$

is undecidable.

Do MSGs G_1 and G_2 describe at least one common MSC?

Do MSGs have an MSC in common?

Theorem: undecidability of empty intersection

The decision problem:

for MSGs G_1 and G_2, do we have $L(G_1) \cap L(G_2) = \varnothing$?

is undecidable.

Proof: Reduction from Post's Correspondence Problem (PCP)

... black board ...

next lecture