

Theoretical Foundations of the UML

Lecture 5: Compositional Message Sequence Graphs

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

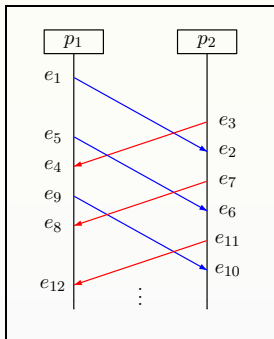
<http://moves.rwth-aachen.de/i2/uml09100/>

16. November 2012

- 1 A non-decomposable MSC
- 2 Compositional Message Sequence Charts
- 3 Compositional Message Sequence Graphs
- 4 Safe Compositional Message Sequence Graphs
- 5 Existence of Safe Paths
- 6 Universality of Safe Paths

- 1 A non-decomposable MSC
- 2 Compositional Message Sequence Charts
- 3 Compositional Message Sequence Graphs
- 4 Safe Compositional Message Sequence Graphs
- 5 Existence of Safe Paths
- 6 Universality of Safe Paths

An MSC that cannot be decomposed [Yannakakis 1999]



This MSC **cannot** be decomposed as

$$M_1 \bullet M_2 \bullet \dots \bullet M_n \quad \text{for } n > 1$$

This can be seen as follows:

- e_1 and $e_2 = m(e_1)$ must **both** belong to M_1
- $e_3 < e_2$ and $e_1 < e_4$ thus
 $e_3, e_4 \notin M_j$, for $j < 1$ or $j > 1$
 $\implies e_3, e_4$ must belong to M_1
- by similar reasoning: $e_5, e_6 \in M_1$ etc.

Problem:

Compulsory matching between send and receive in **same** MSG vertex (i.e., send e and receive $m(e)$ must belong to the same MSC).

- 1 A non-decomposable MSC
- 2 **Compositional Message Sequence Charts**
- 3 Compositional Message Sequence Graphs
- 4 Safe Compositional Message Sequence Graphs
- 5 Existence of Safe Paths
- 6 Universality of Safe Paths

Compositional MSCs [Gunter, Muscholl, Peled 2001]

Solution: drop restriction that e and $m(e)$ belong to the same MSC
(= allow for incomplete message transfer)

Definition (Compositional MSC)

$M = (\mathcal{P}, E, \mathcal{C}, l, m, <)$ is a **compositional MSC** (CMSC, for short) where $\mathcal{P}, E, \mathcal{C}$ and l are defined as before, and

- $m : E_! \rightarrow E_?$ is a **partial, injective** function such that (as before):

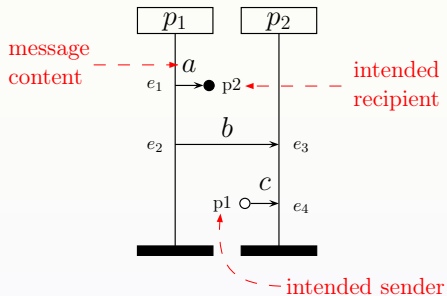
$$m(e) = e' \wedge l(e) = !(p, q, a) \implies l(e') = ?(q, p, a)$$

- $< = \left(\bigcup_{p \in \mathcal{P}} <_p \quad \cup \quad \{(e, m(e)) \mid e \in \underbrace{\text{dom}(m)}_{\text{"m(e) is defined"}}\} \right)^*$

Note:

An MSC is a CMSC where m is total and bijective.

CMSC example



$$\begin{aligned} m(e_2) &= e_3 \\ e_1 &\notin \text{dom}(m) \\ e_4 &\notin \text{rng}(m) \end{aligned}$$

Concatenation of CMSCs (1)

Let $M_i = (\mathcal{P}_i, E_i, \mathcal{C}_i, l_i, m_i, <_i) \in \mathbb{CM}$ $i \in \{1, 2\}$
be CMSCs with $E_1 \cap E_2 = \emptyset$

The **concatenation** of CMSCs M_1 and M_2 is the CMSC
 $M_1 \bullet M_2 = (\mathcal{P}_1 \cup \mathcal{P}_2, E, \mathcal{C}_1 \cup \mathcal{C}_2, l, m, <)$ with:

- $E = E_1 \cup E_2$
- $l(e) = l_1(e)$ if $e \in E_1$, $l_2(e)$ otherwise
- $m(e) = E_1 \rightarrow E_2$ satisfies:
 - ① m extends m_1 and m_2 , i.e., $e \in \text{dom}(m_i)$ implies $m(e) = m_i(e)$
 - ② m matches unmatched send events in M_1 with unmatched receive events in M_2 according to order on process (matching from top to bottom)
the k -th unmatched send in M_1 is matched with the k -th unmatched receive in M_2 (of the same “type”)
 - ③ $M_1 \bullet M_2$ is FIFO (when restricted to matched events)

Concatenation of CMSCs (2)

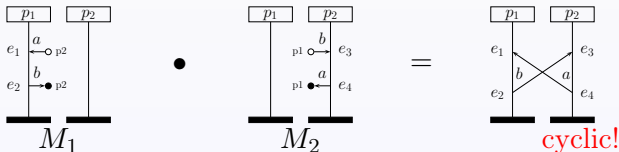
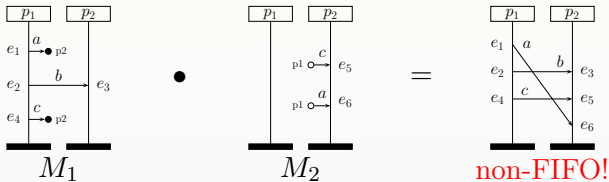
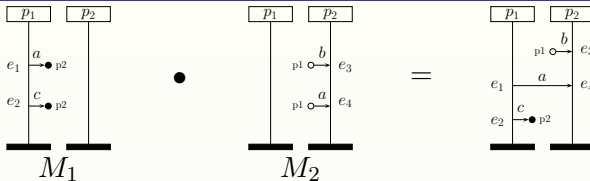
Let $M_i = (\mathcal{P}_i, E_i, \mathcal{C}_i, l_i, m_i, <_i) \in \mathbb{CM}$ $i \in \{1, 2\}$
be CMSCs with $E_1 \cap E_2 = \emptyset$

The **concatenation** of CMSCs M_1 and M_2 is the CMSC
 $M_1 \bullet M_2 = (\mathcal{P}_1 \cup \mathcal{P}_2, E_1 \cup E_2, \mathcal{C}_1 \cup \mathcal{C}_2, l, m, <)$ with:

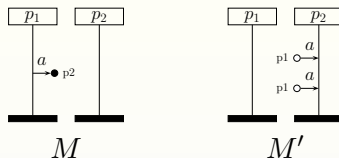
- $<$ is the reflexive and transitive closure of:

$$\begin{aligned} \left(\bigcup_{p \in \mathcal{P}} <_{p,1} \cup <_{p,2} \right) \cup & \{ (e, e') \mid e \in E_1 \cap E_p, e' \in E_2 \cap E_p \} \\ \cup & \{ (e, m(e)) \mid e \in \text{dom}(m) \} \end{aligned}$$

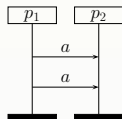
Examples



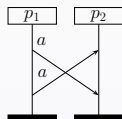
Associativity



$(M \bullet M) \bullet M'$:



$M \bullet (M \bullet M')$:



this is non-FIFO
(and thus undefined)

Note:

Concatenation of CMSCs is not associative.

- 1 A non-decomposable MSC
- 2 Compositional Message Sequence Charts
- 3 Compositional Message Sequence Graphs**
- 4 Safe Compositional Message Sequence Graphs
- 5 Existence of Safe Paths
- 6 Universality of Safe Paths

Let \mathbb{CM} be the set of all CMSCs.

Definition (Compositional MSG)

A **compositional MSG** (CMSG) $G = (V, \rightarrow, v_0, F, \lambda)$ with $\lambda : V \rightarrow \mathbb{CM}$, where V, \rightarrow, v_0 , and F as before.

The difference with an MSG is that the vertices in a CMSG are labeled with compositional MSCs (rather than “real” MSCs).

Paths

Let $G = (V, \rightarrow, v_0, F, \lambda)$ be a CMSG.

Definition (Path in a CMSG)

A **path** π of G is a finite sequence

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

Definition (Accepting path of a CMSG)

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

Definition (CMSC of a path)

The **CMSC of a path** $\pi = u_0 \ \dots \ u_n$ is:

$$M(\pi) = (\dots (\lambda(u_0) \bullet \lambda(u_1)) \bullet \lambda(u_2) \dots) \bullet \lambda(u_n)$$

where CMSC concatenation is left associative.

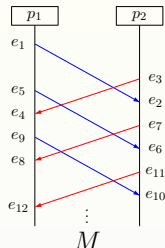
The MSC language of a CMSG

Definition (Language of a CMSG)

The **(MSC) language** of CMSG G is defined by:

$$L(G) = \{ \underbrace{M(\pi) \in \mathbb{M}}_{\text{only "real" MSCs}} \mid \pi \text{ is an accepting path of } G \}.$$

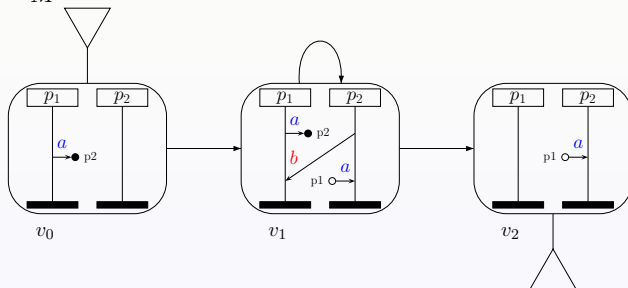
Yannakakis' example as compositional MSG



This MSC cannot be modeled for $n > 1$ by:

$$M = M_1 \bullet M_2 \bullet \dots \bullet M_n \quad \text{with} \quad M_i \in \mathbb{M}$$

But it can be modeled as compositional MSG:



- 1 A non-decomposable MSC
- 2 Compositional Message Sequence Charts
- 3 Compositional Message Sequence Graphs
- 4 Safe Compositional Message Sequence Graphs**
- 5 Existence of Safe Paths
- 6 Universality of Safe Paths

Definition (Safe path)

Path π of CMSG G is **safe** whenever $M(\pi) \in \mathbb{M}$.

Definition (Safe CMSG)

CMSG G is **safe** if for every accepting path π of G , $M(\pi)$ is an MSC.

So:

CMSG G is safe if on any of its accepting paths there are no unmatched sends and receipts, i.e., if any of its accepting runs is indeed an MSC.

- 1 A non-decomposable MSC
- 2 Compositional Message Sequence Charts
- 3 Compositional Message Sequence Graphs
- 4 Safe Compositional Message Sequence Graphs
- 5 Existence of Safe Paths
- 6 Universality of Safe Paths

Existence of a safe accepting path

Theorem: undecidability of existence of a safe path

The decision problem “does CMSG G have at least one safe, accepting path?” is undecidable.

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

... black board ...

- 1 A non-decomposable MSC
- 2 Compositional Message Sequence Charts
- 3 Compositional Message Sequence Graphs
- 4 Safe Compositional Message Sequence Graphs
- 5 Existence of Safe Paths
- 6 Universality of Safe Paths

Universality of safe accepting paths

Theorem: undecidability of existence of a safe path

The decision problem “does CMSG G have **at least one** safe, accepting path?” is **undecidable**.

Theorem: decidability of universality of safe paths

The decision problem “are all accepting paths of CMSG G safe?” is **decidable in PTIME**.

Proof: Polynomial reduction to reachability problem in pushdown automata

... see details on the next slides ...

Definition (Pushdown automaton)

A **pushdown** automaton (PDA, for short) $K = (Q, q_0, \Gamma, \Sigma, \Delta)$ with

- Q , a finite set of control states
- $q_0 \in Q$, the initial state
- Γ , a finite stack alphabet
- Σ , a finite input alphabet
- $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \Gamma^*$, the transition relation.

Example

$(q, a, \gamma, q', \text{pop}) \in \Delta$ means: in state q , on reading input symbol a and top of stack is symbol γ , change to q' and pop γ .

Reachability in pushdown automata

Definition

A **configuration** c is a triple (state q , stack content Z , rest input w).

Definition

Given a transition in Δ , a (direct) **successor** configuration c' of c is obtained: $c \vdash c'$.

Reachability problem

For configuration c , and initial configuration c_0 : $c_0 \vdash^* c$?

Theorem: [Esparza et al. 2000]

The reachability problem for PDA is decidable in PTIME.

Checking whether a CMSG is safe is decidable

- Consider any ordered pair (p_i, p_j) of processes in CMSG G

- Proof idea: construct a PDA $K_{i,j} = (Q, q_0, \Gamma, \Sigma, \Delta)$ such that

CMSG G is not safe wrt. (p_i, p_j) iff PDA $K_{i,j}$ accepts

- For accepting path $u_0 \dots u_k$ in G , feed $K_{i,j}$ with

$$\rho_0 \dots \rho_k \text{ where } \rho_i \in \text{Lin}(\lambda(u_i))$$

- Possible violations that $K_{i,j}$ may encounter:

- 1 nr. of unmatched $!(p_i, p_j, \cdot) >$ nr. of unmatched $?(p_j, p_i, \cdot)$
- 2 type of k -th unmatched send \neq type of k -th unmatched receive
- 3 non-FIFO communication

The nondeterministic PDA $K_{i,j}$

Let $\{a_1, \dots, a_k\}$ be the message contents in CMSG G for (p_i, p_j) .

Nondeterministic PDA $K_{i,j} = (Q, q_0, \Gamma, \Sigma, \Delta)$ where:

- Control states $Q = \{q_0, q_{a_1}, \dots, q_{a_k}, q_{err}, q_F\}$
- Stack alphabet $\Gamma = \{1, \perp\}$
1 counts nr. of unmatched $!(p_i, p_j, a_m)$, and \perp is stack bottom
- Input alphabet $\Sigma = \begin{cases} \text{unmatched action } !(p_i, p_j, a_m) \\ \text{unmatched action } ?(p_j, p_i, a_m) \\ \text{matched actions } !(p_i, p_j, a_m), !(p_j, p_i, a_m) \end{cases}$
- Transition function Δ is described on next slide

Safeness of CMSGs (2)

- Initial configuration is (q_0, \perp, w)
 - w is linearization of actions at p_i and p_j on an accepting path of G
- On reading $!(p_i, p_j, a_m)$ in q_0 , push 1 on stack
 - nondeterministically move to state q_{a_m} or stay in q_0
- On reading $?(p_j, p_i, a_m)$ in q_0 , proceed as follows:
 - if 1 is on stack, pop it
 - otherwise, i.e., if stack is empty, **accept** (i.e., move to q_F)
- On reading matched send $!?(p_i, p_j, a_k)$ in q_0
 - stack empty? ignore input; otherwise, **accept**
- Ignore the input in state q_0 ,
 - on reading matched send events $!?(p_j, p_i, a_k)$, or
 - on reading unmatched sends or receipts not related to p_i and p_j
- Input empty? **Accept**, if stack non-empty; otherwise reject

Safeness of CMSGs (3)

The behaviour in state q_{a_m} for $0 < m \leq k$:

- Ignore all actions except $?(p_j, p_i, a_\ell)$ for all $0 < \ell \leq k$
- On reading $?(p_j, p_i, a_\ell)$ (for some $0 < \ell \leq k$) in state q_{a_m} do:
 - if 1 is on top of stack, pop it
- If stack is empty:
 - if last receive differs from a_m , **accept**
 - otherwise reject, while ignoring the rest (if any) of the input

Safeness of CMSGs (4)

- It follows: PDA $K_{i,j}$ accepts iff CMSG G is not safe wrt. (p_i, p_j)
- \implies CMSG G is not safe wrt. (p_i, p_j) iff configuration (q_F, \cdot, \cdot) is reachable.
- \implies reachability of a configuration in a PDA is in PTIME, hence checking safeness wrt. (p_i, p_j) is in PTIME.

Time complexity

The time complexity of checking whether CMSG G is safe is in $\mathcal{O}(k^2 \cdot N^2 \cdot M \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $M = |\mathcal{C}|$.

Proof.

Checking reachability in PDA $K_{i,j}$ is in $\mathcal{O}(M \cdot |E|^2)$. The number of PDAs is k^2 , as we consider ordered pairs. The number of paths that need to be considered in the CMSG is in $\mathcal{O}(N^2)$, as it suffices to consider a single traversal for each loop in the CMSG. □