Theoretical Foundations of the UML
Lecture 5+6: Compositional Message Sequence Graphs

Joost-Pieter Katoen
Lehrstuhl für Informatik 2
Software Modeling and Verification Group

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

May 4, 2020
Outline

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

motivation

undecidable
decidable

two decision problems
Overview

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

This MSC cannot be decomposed as $M_1 \cdot M_2 \cdot \ldots \cdot M_n$ for $n > 1$.

This can be seen as follows:

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

Problem: Compulsory matching between send and receive events in the same MSG vertex (i.e., send $e$ and receive $m(e)$ must belong to the same MSC).
This MSC cannot be decomposed as

\[ M_1 \circ M_2 \circ \ldots \circ M_n \quad \text{for } n > 1 \]
An MSC that cannot be decomposed

[Yannakakis 1999]

This MSC \textit{cannot} be decomposed as

\[ M_1 \cdot M_2 \cdot \ldots \cdot M_n \quad \text{for } n > 1 \]

This can be seen as follows:

- \( e_1 \) and \( e_2 = m(e_1) \) must \textit{both} belong to \( M_1 \)
This MSC cannot be decomposed as

\[ M_1 \bullet M_2 \bullet \ldots \bullet M_n \quad \text{for } n > 1 \]

This can be seen as follows:

- \( e_1 \) and \( e_2 = m(e_1) \) must both belong to \( M_1 \)
- \( e_3 \preceq e_2 \) and \( e_1 \preceq e_4 \) thus \( e_3, e_4 \notin M_j \), for \( j < 1 \) and \( j > 1 \)
  \[ \implies e_3, e_4 \text{ must belong to } M_1 \]
An MSC that cannot be decomposed

This MSC cannot be decomposed as

\[ M_1 \cdot M_2 \cdot \ldots \cdot M_n \quad \text{for } n > 1 \]

This can be seen as follows:

- \( e_1 \) and \( e_2 = m(e_1) \) must both belong to \( M_1 \)
- \( e_3 \preceq e_2 \) and \( e_1 \preceq e_4 \) thus
  \[ e_3, e_4 \notin M_j \quad \text{for } j < 1 \text{ and } j > 1 \]
  \[ \Rightarrow e_3, e_4 \text{ must belong to } M_1 \]
- by similar reasoning: \( e_5, e_6 \in M_1 \) etc.
An MSC that cannot be decomposed  

This MSC cannot be decomposed as

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

This can be seen as follows:

- $e_1$ and $e_2 = m(e_1)$ must both belong to $M_1$
- $e_3 \preceq e_2$ and $e_1 \preceq e_4$ thus $e_3, e_4 \not\in M_j$, for $j < 1$ and $j > 1$
  $$\implies e_3, e_4 \text{ must belong to } M_1$$
- by similar reasoning: $e_5, e_6 \in M_1$ etc.

**Problem:**

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

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

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

![MSC diagram]

Note: An MSC is a CMSC where $m$ is total and bijective.
Solution: drop restriction that $e$ and $m(e)$ belong to the same MSC (= allow for incomplete message transfer)

**Definition (Compositional MSC)**

$M = (\mathcal{P}, E, C, l, m, \preceq)$ is a compositional MSC (CMSC, for short) where $\mathcal{P}, E, C$ and $l$ are defined as before, and $m : E \rightarrow \{p, q, a\}$ is a partial, injective function such that (as before):

- $m(e) = e_0 \overset{l}{\rightarrow} (p, q, a)$ implies $l(e_0) = ? (q, p, a)$

Note: An MSC is a CMSC where $m$ is total and bijective.
Solution: drop restriction that $e$ and $m(e)$ belong to the same MSC (= allow for incomplete message transfer)

**Definition (Compositional MSC)**

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

- $m : E! \to E?$ is a **partial, injective** function such that (as before):
  
  $m(e) = e' \land l(e) = !(p, q, a)$ implies $l(e') = ?(q, p, a)$

  In MSCs, it is a **bijection**

  In MSCs, it is a **total** function

  **injective** $e_1, e_2 \in E)$:

  $e_1 \neq e_2 \implies m(e_1) \neq m(e_2)$
Solution: drop restriction that \( e \) and \( m(e) \) belong to the same MSC

\( (= \) allow for incomplete message transfer)\n
**Definition (Compositional MSC)**

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

\[ m : E! \rightarrow E? \] is a partial, injective function such that (as before):

\[ m(e) = e' \land l(e) = !(p, q, a) \implies l(e') = ?(q, p, a) \]

\[ \leq = \left( \bigcup_{p \in \mathcal{P}} \lessdot_p \bigcup \{(e, m(e)) \mid e \in \text{dom}(m) \} \right)^* \]

- **Vertical ordering**
- **Horizontal ordering**

Note: An MSC is a CMSC where \( m \) is total and bijective.
Compositional MSCs

[Gunther, Muscholl, Peled 2001]

Solution: drop restriction that \( e \) and \( m(e) \) belong to the same MSC

\( (= \text{ allow for incomplete message transfer}) \)

Definition (Compositional MSC)

\[ M = (P, E, C, l, m, \preceq) \] is a compositional MSC (CMSC, for short) where \( P, E, C \) and \( l \) are defined as before, and

- \( m : E! \to E? \) is a partial, injective function such that (as before):

\[
m(e) = e' \land l(e) = !(p, q, a) \quad \text{implies} \quad l(e') = ?(q, p, a)
\]

- \( \preceq = \left( \bigcup_{p \in P} <_p \right) \cup \{ (e, m(e)) \mid e \in \text{dom}(m) \}^* \)

\( \text{domain of } m \)

\( \text{“} m(e) \text{ is defined} \)
CMSC example

- \( m(e_2) = e_3 \)
- \( e_1 \notin \text{dom}(m) \)
- \( e_4 \notin \text{rng}(m) \)

\[
E_1 = \{ e_1, e_2 \} \\
E_2 = \{ e_3, e_4 \} \\
\neg \exists e. m(e) = e_4
\]
Let $M_i = (P_i, E_i, C_i, l_i, m_i, \preceq_i) \in \text{CM} \quad i \in \{1, 2\}$
be CMSCs with $E_1 \cap E_2 = \emptyset$.

The concatenation of CMSCs $M_1$ and $M_2$ is the CMSC $M_1 \cdot M_2 = (P_1 \cdot P_2, E, C_1 \cdot C_2, \underbrace{l_1 \cdot l_2}_{i} m_1 \cdot m_2)$
with:

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

1. $m$ extends $m_1$ and $m_2$, i.e., $e_2 \in \text{dom}(m_i)$ 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 ($\text{matching from top to bottom}$)

3. $M_1 \cdot M_2$ is FIFO (when restricted to matched events)
Let $M_i = (\mathcal{P}_i, E_i, \mathcal{C}_i, l_i, m_i, \preceq_i) \in \text{CM} \quad i \in \{1, 2\}$ be CMSCs with $E_1 \cap E_2 = \emptyset$

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

1. $E$ extends $E_1$ and $E_2$, i.e., $e \in E$ means that $e \in E_1$ or $e \in E_2$.
2. $l$ extends $l_1$ and $l_2$, i.e., $e \in l$ means that $e \in l_1$ or $e \in l_2$.
3. $m$ extends $m_1$ and $m_2$, i.e., $e \in m$ means that $e \in m_1$ or $e \in m_2$.
4. $\preceq$ extends $\preceq_1$ and $\preceq_2$, i.e., $e_1 \preceq e_2$ means that $e_1 \preceq_1 e_2$ or $e_1 \preceq_2 e_2$.
Let $M_i = (P_i, E_i, C_i, l_i, m_i, \preceq_i) \in \mathcal{CM}$, $i \in \{1, 2\}$ be CMSCs with $E_1 \cap E_2 = \emptyset$.

The concatenation of CMSCs $M_1$ and $M_2$ is the CMSC $M_1 \bullet M_2 = (P_1 \cup P_2, E, C_1 \cup C_2, l, m, \preceq)$ with:

- $E = E_1 \cup E_2$
- $l(e) = l_1(e)$ if $e \in E_1$, $l_2(e)$ otherwise

$\{ \text{all the same as for concatenation of MSCs.} \}$
Let $M_i = (P_i, E_i, C_i, l_i, m_i, \preceq_i) \in \mathbb{CM} \quad i \in \{1, 2\}$ be CMSCs with $E_1 \cap E_2 = \emptyset$

The concatenation of CMSCs $M_1$ and $M_2$ is the CMSC $M_1 \cdot M_2 = (P_1 \cup P_2, E, C_1 \cup C_2, l, m, \preceq)$ with:

- $E = E_1 \cup E_2$
- $l(e) = l_1(e)$ if $e \in E_1$, $l_2(e)$ otherwise
- $m(e) = E! \rightarrow E?$ satisfies:
  - $m$ extends $m_1$ and $m_2$, i.e., $e \in \text{dom}(m_i)$ implies $m(e) = m_i(e)$
  - For events in $E_1$ for which $m_i$ is defined, the matching event remains the same.
Concatenation of CMSCs (1)

Let $M_i = (P_i, E_i, C_i, l_i, m_i, \preceq_i) \in \mathcal{CM}$ \quad $i \in \{1, 2\}$

be CMSCs with $E_1 \cap E_2 = \emptyset$

The concatenation of CMSCs $M_1$ and $M_2$ is the CMSC $M_1 \cdot M_2 = (P_1 \cup P_2, E, C_1 \cup C_2, l, m, \preceq)$ with:

- $E = E_1 \cup E_2$
- $l(e) = l_1(e)$ if $e \in E_1$, $l_2(e)$ otherwise
- $m(e) = E! \rightarrow E?$ 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)
Let \( M_i = (P_i, E_i, C_i, l_i, m_i, \preceq_i) \in \mathbb{CM} \quad i \in \{1, 2\} \) be CMSCs with \( E_1 \cap E_2 = \emptyset \)

The concatenation of CMSCs \( M_1 \) and \( M_2 \) is the CMSC
\( M_1 \bullet M_2 = (P_1 \cup P_2, E, C_1 \cup C_2, l, m, \preceq) \) with:

- \( E = E_1 \cup E_2 \)
- \( l(e) = l_1(e) \) if \( e \in E_1 \), \( l_2(e) \) otherwise
- \( m(e) = E! \rightarrow E? \) 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”)

\[ \text{same message content} \]
\[ \text{sender corresponds to receiver} \]
Concatenation of CMSCs (1)

Let $M_i = (P_i, E_i, C_i, l_i, m_i, \preceq_i) \in \text{CM} \quad i \in \{1, 2\}$ be CMSCs with $E_1 \cap E_2 = \emptyset$

The concatenation of CMSCs $M_1$ and $M_2$ is the CMSC $M_1 \bullet M_2 = (P_1 \cup P_2, E, C_1 \cup C_2, l, m, \preceq)$ with:

- $E = E_1 \cup E_2$
- $l(e) = l_1(e)$ if $e \in E_1$, $l_2(e)$ otherwise
- $m(e) = E! \rightarrow E?$ 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)
  3. the $k$-th unmatched send in $M_1$ is matched with the $k$-th unmatched receive in $M_2$ (of the same “type”)
  4. $M_1 \bullet M_2$ is FIFO (when restricted to matched events)
Let $M_i = (P_i, E_i, C_i, l_i, m_i, \preceq_i) \in \text{CM} \quad i \in \{1, 2\}$ be CMSCs with $E_1 \cap E_2 = \emptyset$

The concatenation of CMSCs $M_1$ and $M_2$ is the CMSC $M_1 \cdot M_2 = (P_1 \cup P_2, E_1 \cup E_2, C_1 \cup C_2, l, m, \preceq)$ with:
Concatenation of CMSCs (2)

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

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

- \( l \) and \( m \) are defined as on the previous slide.
- \( \preceq \) is the reflexive and transitive closure of:

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

process order

process wise: all events at process \( p \) in \( M_2 \) happen after all events at \( p \) in \( M_1 \).
Examples

(1) \[ M_1 \rightarrow M_2 \]

(2) \[ M_1 \rightarrow M_2 \]

(3) \[ M_1 \rightarrow M_2 \]

For this \( M_1 \) and \( M_2 \), \( M_1 \cdot M_2 \) is not defined.
Associativity

1. \((M \cdot M) \cdot M'\):
   \[
   \begin{array}{c}
   p_1 \quad p_2 \\
   a \quad p_2 \\
   \end{array}
   \]
   \[
   \begin{array}{c}
   p_1 \quad p_2 \\
   a \quad p_1 \\
   \end{array}
   \]

2. \(M \cdot (M \cdot M')\):
   \[
   \begin{array}{c}
   p_1 \quad p_2 \\
   a \quad \quad a \\
   \end{array}
   \]
   \[
   \begin{array}{c}
   p_1 \quad p_2 \\
   a \quad \quad a \\
   \end{array}
   \]

For MSCs, \(\cdot\) is associative.

\[(M_1 \cdot M_2) \cdot M_3 = M_1 \cdot (M_2 \cdot M_3)\]

This is non-FIFO (and thus undefined).

Note: Concatenation of CMSCs is not associative.
Associativity

Note:
Concatenation of CMSCs is not associative.
Overview

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 $\mathcal{CM}$ be the set of all CMSCs.

**Definition (Compositional MSG)**

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

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

Let $G = (V, E, v_0, F)$ be a CMSG.

Definition (Path in a CMSG)
A path $\pi$ of $G$ is a finite sequence $\pi = u_0 u_1 \ldots u_n$ with $u_i \in V (0 \leq i \leq n)$ and $u_i \neq u_{i+1} (0 \leq i < n)$.

Definition (Accepting path of a CMSG)
Path $\pi = u_0 \ldots u_n$ is accepting if $u_0 = v_0$ and $u_n \in F$.

Definition (CMSC of a path)
The CMSC of a path $\pi = u_0 \ldots u_n$ is:
$$M(\pi) = (\ldots (M(u_0) \cdot M(u_1)) \cdot M(u_2) \ldots \cdot M(u_n))$$
where CMSC concatenation is left associative.
Let $G = (V, \rightarrow, v_0, F, \lambda)$ be a CMSG.
Paths

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

Definition (Path in a CMSG)

A path $\pi$ of $G$ is a finite sequence

$\pi = u_0 \ u_1 \ \ldots \ u_n$ with $u_i \in V \ (0 \leq i \leq n)$ and $u_i \rightarrow u_{i+1} \ (0 \leq i < n)$
Let \( G = (V, \rightarrow, v_0, F, \lambda) \) be a CMSG.

**Definition (Path in a CMSG)**

A path \( \pi \) of \( G \) is a finite sequence

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

**Definition (Accepting path of a CMSG)**

Path \( \pi = u_0 \ldots u_n \) is accepting if: \( u_0 = v_0 \) and \( u_n \in F \).
Paths

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

**Definition (Path in a CMSG)**

A path \( \pi \) of \( G \) is a finite sequence

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

**Definition (Accepting path of a CMSG)**

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

**Definition (CMSC of a path)**

The CMSC of a path \( \pi = u_0 \ldots u_n \) is:

\[
M(\pi) = (\ldots (\lambda(u_0) \cdot \lambda(u_1)) \cdot \lambda(u_2) \ldots) \cdot \lambda(u_n)
\]

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:

$$L(G) = \{ M(\pi) \in M \mid \pi \text{ is an accepting path of } G \}.$$
The MSC language of a CMSG

Definition (Language of a CMSG)

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

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

Note: Accepting paths that give rise to an CMSC (which is not an MSC) are not part of $L(G)$. 

Joost-Pieter Katoen

Theoretical Foundations of the UML
Yannakakis’ example as compositional MSG

This MSC cannot be modeled for $n \geq 1$ by:

$$M = M_1 \cdot M_2 \cdot ... \cdot M_n$$

Thus it cannot be modeled by a MSG.

But it can be modeled as compositional MSG.
This MSC cannot be modeled for $n > 1$ by:

$$M = M_1 \cdot M_2 \cdot \ldots \cdot M_n \quad \text{with} \quad M_i \in \mathbb{M}$$
Yannakakis’ example as compositional MSG

This MSC cannot be modeled for \( n > 1 \) by:

\[
M = M_1 \cdot M_2 \cdot \ldots \cdot M_n \quad \text{with} \quad M_i \in \mathbb{M}
\]

Thus it cannot be modeled by a MSG.
Yannakakis’ example as compositional MSG

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

$$M = M_1 \cdot M_2 \cdot \ldots \cdot M_n \quad \text{with} \quad M_i \in \mathbb{M}$$

Thus it cannot be modeled by a MSG. But it can be modeled as **compositional** MSG:
CMS: $g$:

MSC $M_1 \in L(G)$

Every accepting path $\pi$ for $G$: $M(\pi)$ is an MSC

$\Rightarrow M(\pi) \in L(G)$

CMS $g$ is called safe
Overview

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
Safe paths and CMSGs

Definition (Safe path)
A path $\pi$ of a CMSG $G$ is safe whenever $M(\pi) \in M$.

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

So:
A 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 paths is indeed an MSC.
Safe paths and CMSGs

Definition (Safe path)
Path $\pi$ of CMSG $G$ is safe whenever $M(\pi) \in M$.

MSC

in the CMSC of $\pi$
Definition (Safe path)
Path $\pi$ of CMSG $G$ is safe whenever $M(\pi) \in M$.

Definition (Safe CMSG)
CMSG $G$ is safe if for every accepting path $\pi$ of $G$, $M(\pi)$ is an MSC.
Safe paths and CMSGs

Definition (Safe path)
Path $\pi$ of CMSG $G$ is safe whenever $M(\pi) \in M$.

Definition (Safe CMSG)
CMSG $G$ is safe if for every accepting path $\pi$ 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 paths is indeed an MSC.
Overview

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
Theorem: undecidability of existence of a safe path

The decision problem “does CMSG $G$ have at least one safe, accepting path?” is undecidable.
Theorem: undecidability of existence of a safe path

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

Proof.

By a reduction from Post’s Correspondence Problem (PCP).

... black board ...
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.

By a reduction from Post’s Correspondence Problem (PCP).

... black board ...

The complement decision problem “does CMSG $G$ have no safe, accepting path?” is undecidable too.
Claim: the decision problem “does $CMS G G$ have at least one safe path?” is undecidable.

Proof by a reduction from the PCP problem.

Proof idea: instance of PCP $\rightarrow$ instance $CMS G G_{u,w}$

$\{(u,w)\}$

$U = \{u_1, \ldots, u_n\}$ $w_i \in \Sigma^*$

$W = \{w_1, \ldots, w_n\}$ $w_i \in \Sigma^*$

such that $(u,w)$ has a solution if and only if $CMS G G_{u,w}$ has a safe, accepting path.

$s_j \in [1..n]$ such that $u_1 u_2 \ldots u_n = w_1 w_2 \ldots w_n$

How does the $CMS G G_{u,w}$ look like?
Components of CMSG $G_{u,w}$:

$$P = \{p_1, p_2, p_3, p_4\}$$ processes

$$C = \{c_{1, \ldots, n}\}$$

$$V = \{v_1, \ldots, v_n\} \cup \{\tilde{v}_1, \ldots, \tilde{v}_m\} \cup \{v_F\}$$

$$F = \{v_F\}$$

$$\lambda(v_i) = \text{CMSG corresponding to the word } u_i$$

$$\lambda(v'_i) = \omega$$

How do the vertices $V_i$, $V'_i$ and $v_F$ look like?

By example, let $\Sigma = \{a, b\}$, $u_i = \text{abaa}$, $u_F = \text{ba}$.

Then:

$$\lambda(v_i) = \begin{array}{cccc}
p_1 & p_2 & p_3 & p_4 \\
\text{a} & \text{b} & \text{a} & \text{b} \\
\text{b} & \text{a} & \text{a} & \text{b} \\
\text{a} & \text{a} & \text{a} & \text{a} \\
\end{array}$$

"send the word $u_i$ and index $i$"

$$\lambda(v'_i) = \begin{array}{cccc}
p_1 & p_2 & p_3 & p_4 \\
\text{b} & \text{a} & \text{a} & \text{b} \\
\text{a} & \text{a} & \text{a} & \text{a} \\
\text{a} & \text{a} & \text{a} & \text{a} \\
\end{array}$$

"receive the word $u'_i$ plus index $i"
\( \lambda (v_F) \)

\[ \begin{array}{cccc}
  \text{end} & \text{end} \\
  \text{①} & \text{②} \\
  \text{③} & \\
\end{array} \]

① indicates that process \( p_1 \) has sent all its messages to \( p_2 \) and if ① is received by \( p_2 \), all messages of \( p_1 \) have been received by \( p_2 \).

② similar as ① but now for the “index” messages that are exchanged between \( p_3 \) & \( p_4 \).

③ indicates that both “phases” ① and ② have finished.
It remains to prove that the reduction:

\[ \text{PCP instance } (U, W) \rightarrow \text{CMSG } G_{U,W} \]

is correct. That is, our proof obligation is:

\[ (U, W) \text{ has a solution iff } G_{U,W} \text{ has a safe, accepting path} \]

**Proof:** "\( \Rightarrow \)" let index sequence \( i_1, \ldots, i_k \) be a solution of PCP instance \((U, W)\). Then there is an accepting path in \( G_{U,W} \):

\[ \Pi = V_{i_1} \ldots V_{i_k} \]

As \( i_1, \ldots, i_k \) is a solution to \((U, W)\), and by construction of the CMSG \( G_{U,W} \) it follows that:

\[ M(\Pi) = \left( \left( \lambda(V_{i_1}) \cdot \ldots \right) \cdot \lambda(V_{i_k}) \right) \cdot \ldots \cdot \lambda(V_{i_k}) \cdot \lambda(V_F) \]

(left-associated bracketing) is an MSC.

Thus \( \Pi \) is safe and \( \Pi \) is accepting.
Let $\Pi$ be a safe, accepting path in $G_{u,w}$.

Assume:

$$\Pi = V_{i_1} \ldots V_{i_m} \quad V_{j_1} \ldots V_{j_k} \quad VF$$

$m$ steps $k$ steps

with $i_1, \ldots, i_m \in \{1, \ldots, n\}$ and $j_1, \ldots, j_k \in \{1, \ldots, n\}$. Since $\Pi$ is safe and ends in vertex $VF$, it follows:

1. As $(p_4, p_3, \text{end})$ occurs in $VF$, all unmatched sends by $p_3$ in subpath $V_{i_1} \ldots V_{i_m}$ are matched by corresponding receive events by $p_4$ in the subpath $V_{j_1} \ldots V_{j_k}$. As in each vertex $V_{ij}$ one message is sent from $p_3$ and in $V_{jk}$ one message is received by $p_4$, it follows that $m = k$.

2. As $\Pi$ is safe, it follows that $V_{i_1} \ldots V_{i_m} \quad V_{j_1} \ldots V_{j_m}$ is safe and FIFO.

Thus all "index" messages $i_1, \ldots, i_m$ sent by $p_3$ are received by $p_4$, in the same order.
Thus \( c_1 = \hat{c}_1, \ c_2 = \hat{c}_2, \ldots, \ c_m = \hat{c}_m \)

So \( \bar{T}_I = V_{c_1} \cdots V_{c_m} V_{c_1}^' \cdots V_{c_m}^' V_F \) is safe and accepting.

As \( \bar{T}_I \) is completed by \( ?(P_4, P_2, \text{end}) \) and

\! (P_2, P_4, \text{end}) after \( ?(P_2, P_1, \text{end}) \), it follows that

once \( P_4 \) has received all "index" messages, \( P_2 \) has

received all messages sent by \( P_1 \) in \( V_{c_1} \cdots V_{c_m} \).

Process \( P_1 \) has sent \( U_{c_1} \cdots U_{c_m} \) (to \( P_2 \)),

process \( P_2 \) has received \( W_{c_1} \cdots W_{c_m} \).

Since \( \bar{T}_I \) is safe, it follows \( U_{c_1} \cdots U_{c_m} = W_{c_1} \cdots W_{c_m} \).

Thus: \( \bar{U}_1 \cdots \bar{U}_m \) is a solution to the PCP instance \((U, W)\).
<table>
<thead>
<tr>
<th>Overview</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A non-decomposable MSC</td>
<td>✓</td>
<td>Yannakakis</td>
</tr>
<tr>
<td>2. Compositional Message Sequence Charts</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>3. Compositional Message Sequence Graphs</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>4. Safe Compositional Message Sequence Graphs</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>5. Existence of Safe Paths</td>
<td></td>
<td>undecided</td>
</tr>
<tr>
<td>6. Universality of Safe Paths</td>
<td></td>
<td>does a CMSG only have safe paths?</td>
</tr>
</tbody>
</table>