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

{ two decision problems

   undecidable \quad \text{decidable} }
Compositional MSCs

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

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

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

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

Note:
An MSC is a CMSC where $m$ is total and bijective.
CMSC example

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

$E_1 = \{ e_1, e_2 \}$
$E_2 = \{ e_3, e_4 \}$

$\neg \exists e. \ m(e) = e_4$

$m(e_1)$ is not defined
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 $u_i \rightarrow u_{i+1} \ (0 \leq i < n)$

**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:

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

where CMSC concatenation is left associative.
The MSC language of a CMSG

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

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

only “real” MSCs

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
\[
\begin{align*}
\text{accepting path: } & \quad \pi = u_0 u_1 \\
M(\pi) & \in M \\
\text{thus } & \quad M(\pi) \in L(G)
\end{align*}
\]

\[
\begin{align*}
\text{accepting path: } & \quad \pi' = u_0 u_1 u_1 \\
M(\pi') & \notin M \\
M(\pi') & \notin L(G)
\end{align*}
\]
Yannakakis’ example as compositional MSG

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

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

Thus it cannot be modeled by a MSG. But it can be modeled as **compositional** MSG:
Every accepting path \( \pi \) for \( G \): \( M(\pi) \) is an MSC \( \rightarrow M(\pi) \in L(G) \)

CMSG \( g \) is called **safe**
Safe paths and CMSGs

**Definition (Safe path)**

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

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.
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.
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.

Theorem: decidability of universality of safe paths

The decision problem “are all accepting paths of CMSG $G$ safe?” is decidable in PTIME.
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 (non-deterministic) 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
- $\Gamma$, a finite stack alphabet
- $\Sigma$, a finite input alphabet
- $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \Gamma^*$, the transition relation.
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
- \( \Gamma \), a finite stack alphabet
- \( \Sigma \), a finite input alphabet
- \( \Delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \Gamma^* \), the transition relation.

Transition relation

\((q, a, \gamma, q', \text{pop}) \in \Delta \) means: in state \( q \), on reading input symbol \( a \) and top of stack is symbol \( \gamma \), change to \( q' \) and pop \( \gamma \) from the stack.
\[
L = \{0^n1^n \mid n > 0\} \\
\text{Construct a PDA } K \text{ such that } K \text{ accepts the language } L
\]

Intuition

- PDA \( K \) starts in initial control state \( q_0 \)
- if input word \( w = \epsilon \) or if \( w \) starts with a "1" : reject
- otherwise, "scan" all Os and push them on the stack
- on reading the first "1", move to control state \( q_1 \)
  - pop 0 from the stack
- in \( q_1 \), on reading a "1", we pop a '0' from the stack
- in \( q_1 \): reject if a "0" is read, or if input word is <
  - but the stack is not
  - nr. of Os > 1s
- in \( q_1 \): accept if input word and
  - the stack are both empty.
Example configurations:

\[(q_0, 0011, \#), \quad (q, 1, 0)\]

Change of configuration:

\[(q_0, 0011, \#) \rightarrow (q_0, 011, 0) \rightarrow (q_0, 11, 00) \rightarrow (q, 1, 0) \rightarrow (q_1, \$, \$)\]

Is \((q_1, \$, \$)\) reachable from \((q_0, 0011, \#)\)?
Reachability in pushdown automata

Definition

A configuration \( c \) is a triple (state \( q \), stack content \( Z \), rest input \( w \)).
Reachability in pushdown automata

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

Definition
Given a transition in $\Delta$, a (direct) successor configuration $c'$ of $c$ is obtained: $c \vdash c'$.
Reachability in pushdown automata

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

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

**Reachability problem**
For configuration $c$, and initial configuration $c_0$, is $c_0 \vdash^* c$?
Reachability in pushdown automata

Definition

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

Definition

Given a transition in $\Delta$, 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.
Remark: Dyck language

\[ \Sigma = \{ [ , ] \} \] square brackets

Dyck language

\[ \{ u \in \Sigma^* \mid \text{all prefixes of } u \text{ contain no more } [ \text{ than } ] \}, \text{ and the number of } \]

linearization

\[ [ \text{ equals the number of } ] \text{ in } u \]

Exercise construct a PDA that accepts the
dyck language.
$(a_0, \text{00111}, \#) \rightarrow (a_0, \text{0011}, 0)$

$\not\rightarrow L$

$\rightarrow (a_0, \text{011}, 00)$

$\rightarrow (a_0, \text{11}, 000)$

$\rightarrow (a_1, 1, 00)$

$\rightarrow (a_1, 2, 0)$ “reject”

Sometimes “reject” is modeled explicitly by a separate control state $q_{err};$ similarly “accept” by a control state $q_{F}$.

In our example this means that there are transitions:

$(a_0, 1, \#, q_{err}, \#)$

$(a_1, 0, \#, q_{err}, \#)$

$(a_1, 0, 0, q_{err}, \#)$

etcetera.
Checking whether a CMSG is safe is decidable

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

**Proving Idea:** Construct a PDA \(K_{i,j} = (Q, q_0, \Delta, \Gamma)\) such that CMSG \(G\) is not safe wrt. \((p_i, p_j)\) if PDA \(K_{i,j}\) accepts.

For accepting path \(u_0 \ldots u_k\) in \(G\), feed \(K_{i,j}\) with the word \(\Gamma_0 \ldots \Gamma_k\) where \(\Gamma_i \in \text{Lin}(u_i)\) such that unmatched sends (of some type) precede all unmatched receipts (of the same type).

Possible violations that \(K_{i,j}\) may encounter:

1. \(\text{nr. of unmatched} (p_i, p_j, \cdot) > \text{nr. of unmatched} (p_j, p_i, \cdot)\)
2. Type of \(k\)-th unmatched send \(\neq\) type of \(k\)-th unmatched receive
3. Non-FIFO communication

Joost-Pieter Katoen
Theoretical Foundations of the UML
25/29
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

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

- In fact "not left-closed"

- \(K_{i,j}\) is going check for possible violations of being safe
Definition (left-closed CMSC)

A CMSC is left-closed if it does not contain unmatched receive events, or any send events that are not yet matched and precede other matched send events (of the same type).

Examples

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

is not left-closed (unsafe)
accept

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

is not left-closed (unsafe)
accept

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

is left-closed (and safe)
reject

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
</tr>
</tbody>
</table>

is left-closed (not-safe)
reject
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

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

For accepting path \(u_0 \ldots u_k\) in \(G\), feed \(K_{i,j}\) with all words

\[
\rho_0 \ldots \rho_k \text{ where } \rho_i \in \text{Lin}(\lambda(u_i))
\]

such that unmatched sends (of some type) precede all unmatched receipts (of the same type)

\[\text{assume that matched events are indicated explicitly!}\]

\[\text{not a restriction, because such linearisations do always exist! (p,q,a)}\]
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

\[
\text{CMSG } G \text{ is not } \mathcal{L}(\mathcal{C}) \text{ wrt. } (p_i, p_j) \iff \text{PDA } K_{i,j} \text{ accepts}
\]

- For accepting path \(u_0 \ldots u_k\) in \(G\), feed \(K_{i,j}\) with \(\text{all words } \rho_0 \ldots \rho_k\) where \(\rho_i \in \text{Lin}(\lambda(u_i))\)

  such that unmatched sends (of some type) precede all unmatched receipts (of the same type)

- 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
Let \( \{a_1, \ldots, a_k\} \) be the message contents in CMSG \( G \) for \((p_i, p_j)\).
The nondeterministic PDA $K_{i,j}$

Let \( \{a_1, \ldots, a_k\} \) 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 $Q = \{q_0, q_{a_1}, \ldots, q_{a_k}, q_{err}, q_F\}$

Control states:
- Initial
- Reject
- Accept
Let \( \{a_1, \ldots, a_k\} \) 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 \( Q = \{q_0, qa_1, \ldots, qa_k, q_{err}, q_F\} \)
- Stack alphabet \( \Gamma = \{1, \#\} \)
  - 1 counts nr. of unmatched \(! (p_i, p_j, a_m)\), and \# is bottom of stack
The nondeterministic PDA $K_{i,j}$

Let $\{a_1, \ldots, a_k\}$ 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 $Q = \{q_0, qa_1, \ldots, qa_k, q_{err}, q_F\}$

- Stack alphabet $\Gamma = \{1, \#\}$
  
  1 counts nr. of unmatched $!(p_i, p_j, a_m)$, and $\#$ is bottom of stack

- Input alphabet $\Sigma$

  $\Sigma = \begin{cases} 
  \text{unmatched action } !(p_i, p_j, a_m) & \checkmark \\
  \text{unmatched action } ?(p_j, p_i, a_m) & \checkmark \\
  \text{matched actions } !?(p_i, p_j, a_m) & \checkmark 
  \end{cases}$
The nondeterministic PDA $K_{i,j}$

Let $\{a_1, \ldots, a_k\}$ 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** $Q = \{q_0, q_{a_1}, \ldots, q_{a_k}, q_{err}, q_F\}$

- **Stack alphabet** $\Gamma = \{1, \#\}$
  
  1 counts nr. of unmatched $(p_i, p_j, a_m)$, and $\#$ is bottom of stack

- **Input alphabet**
  
  $\Sigma = \begin{cases} 
  \text{unmatched action } !(p_i, p_j, a_m) \\
  \text{unmatched action } ?(p_j, p_i, a_m) \\
  \text{matched action } !? (p_i, p_j, a_m)
  \end{cases}$

- **Transition function** $\Delta$ is described on next slide
Initial configuration is \((q_0, \#, w)\)
- \(w\) is linearization of actions at \(p_i\) and \(p_j\) on an accepting path of \(G\)
Safeness of CMSGs (2)

- Initial configuration is \((q_0, \#, w)\)
  - \(w\) 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 \(q_{am}\) or stay in \(q_0\)

"seen an unmatched send from \(p_i\) to \(p_j\) with content \(a_m\)"
Safeness of CMSGs (2)

- Initial configuration is \((q_0, \#, w)\)
  - \(w\) 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 \(q_F\))

≠ left-closed
Safeness of CMSGs (2)

- Initial configuration is \((q_0, \#, w)\)
  - \(w\) 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 (i.e., equal to \(#\))? ignore input; otherwise, accept

\[\text{not left-clawed}\]
Safeness of CMSGs (2)

- Initial configuration is \((q_0, \#, w)\)
  - \(w\) 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 (i.e., equal to \(\#\))? ignore input; otherwise, accept

- Ignore the following inputs in state \(q_0\):
  - matched send events \(?!(p_j, p_i, a_k)\), and
  - unmatched sends or receipts not related to \(p_i\) and \(p_j\)
Safeness of CMSGs (2)

- Initial configuration is \((q_0, \#, w)\)
  - \(w\) 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 (i.e., equal to \(\#\))? ignore input; otherwise, accept

- Ignore the following inputs in state \(q_0\):
  - matched send events \(!? (p_j, p_i, a_k)\), and
  - unmatched sends or receipts not related to \(p_i\) and \(p_j\)

- Remaining input \(w\) empty? Accept, if stack non-empty; else reject
Safeness of CMSGs (3)

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

- Ignore all actions except $? (p_j, p_i, a_\ell)$ for all $0 < \ell \leq k$
The behaviour in state $q_{am}$ 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 $q_{am}$ do:
  - if 1 is on top of stack, pop it
Safeness of CMSGs (3)

The behaviour in state $q_{am}$ 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 $q_{am}$ 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
Example 1.

CMSG $G_1$

\[ (q_0, \#, !a??a?\text{a}) \xrightarrow{T} (q_a, 1, ?a?a) \]

Initial configuration

\[ (q_a, \#, ?a) \]

accept

Thus the PDA $k_{1,2}$ accepts the input word $\Rightarrow G_1$ is not left-closed
Example 2

PDA $k_{1,2}$

$$(a_0, \#, !a!b ?a?b) \xrightarrow{T} (a_a, 1, !b ?a?b)$$

$$(a_0, 1, !b ?a?b) \xrightarrow{T} (a_a, 1, ?a?b)$$

$$(a_0, 1, ?b) \xrightarrow{T} (a_a, #, ?b)$$

$$(a_0, #, \epsilon) \xrightarrow{T} (a_6, #, \epsilon)$$

Reject

$$L(G) = \frac{1}{b}$$

$\text{PDA } k_{1,2} \text{ has no accepting run on } !a!b ?a?b$$

and thus $\text{CMSG } g_2 \text{ is left-closed}$
Example

PDA $k_{1,2}$

$(q_0, \#, 1a, \#b, \#a)$ \rightarrow $(q_0, 1, 1a, \#a)$

accept

$(q_a, 1, 1a)$

$\Rightarrow$ violation of FIFO property.

$(q_a, 1, \#a)$

PDA $k_{1,2}$ accepts

$(q_a, \#, \varepsilon)$

reject

$\Rightarrow$ CMSG $g_3$ is not left-closed

$\Rightarrow$ is not safe.
Safeness of CMSGs (4)

It follows: PDA $K_{i,j}$ accepts iff CMSG $G$ is not $\mathcal{L}_c$ wrt. $(p_i, p_j)$

$\implies$ CMSG $G$ is not $\mathcal{L}_c$ wrt. $(p_i, p_j)$ iff configuration $(q_F, \cdot, \cdot)$ is reachable.
Safeness of CMSGs (4)

It follows: PDA $K_{i,j}$ accepts iff CMSG $G$ is not $\mathcal{L}$ wrt. $(p_i, p_j)$

$\implies$ CMSG $G$ is not $\mathcal{L}$ 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.

Joost-Pieter Katoen

Theoretical Foundations of the UML
Safeness of CMSGs (4)

It follows: PDA $K_{i,j}$ accepts iff CMSG $G$ is not $\mathcal{L}$ wrt. $(p_i, p_j)$

$\implies$ CMSG $G$ is not $\mathcal{L}$ 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 worst-case time complexity of checking whether CMSG $G$ is safe is in $O(k^2 \cdot N^2 \cdot L \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $L = |C|$.
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 worst-case time complexity of checking whether CMSG $G$ is safe is in $O(k^2 \cdot N^2 \cdot L \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $L = |\mathcal{C}|$.

**Proof.**

Checking reachability in PDA $K_{i,j}$ is in $O(L \cdot |E|^2)$. 
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 worst-case time complexity of checking whether CMSG $G$ is safe is in $O(k^2 \cdot N^2 \cdot L \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $L = |\mathcal{C}|$.

**Proof.**

Checking reachability in PDA $K_{i,j}$ is in $O(L \cdot |E|^2)$. The number of PDAs is $k^2$, as we consider ordered pairs in $\mathcal{P}$. 
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 worst-case time complexity of checking whether CMSG $G$ is safe is in $O(k^2 \cdot N^2 \cdot L \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $L = |\mathcal{C}|$.

Proof.

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