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 5, 2020

Outline

two decision poblems

undecidable

decidable

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)

$M = (\mathcal{P}, E, \mathcal{C}, l, m, \preceq)$ 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):

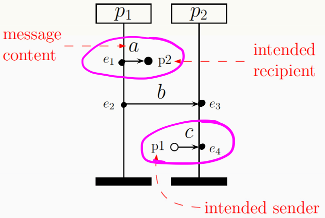
$$m(e) = e' \wedge l(e) = !(p, q, a) \quad \text{implies} \quad l(e') = ?(q, p, a)$$

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

Note:

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

CMSC example



- $m(e_2) = e_3$ ✓
- $e_1 \notin dom(m)$
- $e_4 \notin rng(m)$ ✓

m(e_1) is not defined

$E_! = \{e_1, e_2\}$

$E_? = \{e_3, e_4\}$

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

Paths

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

$\lambda : V \longrightarrow CM$

Definition (Path in a CMSG)

A path π of G is a finite sequence

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

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.

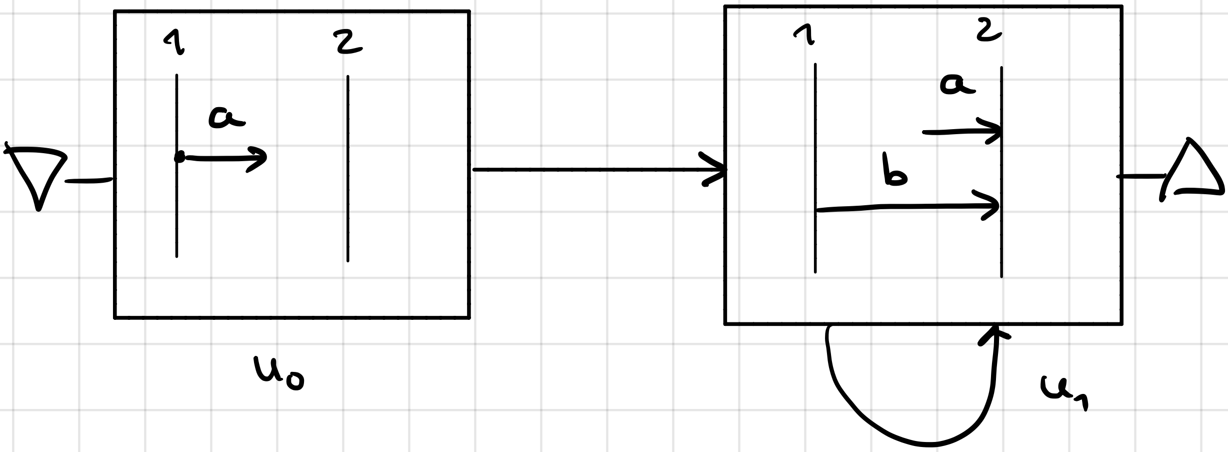
The MSC language of a CMSG

Definition (Language of a CMSG)

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

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

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

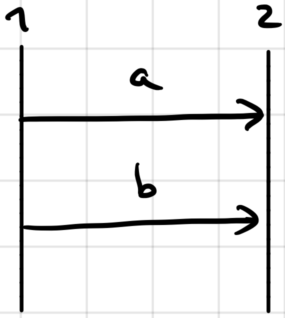


CMSG $\mathcal{G}$

accepting path: $\pi = u_0 u_1$

$M(\pi) \in \mathbb{M}$

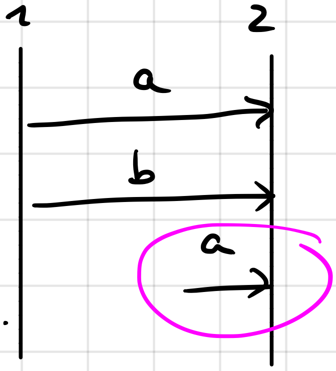
thus $M(\pi) \in L(\mathcal{G})$



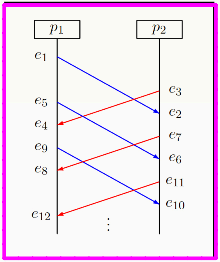
accepting path $\pi' = u_0 u_1 u_1$

$M(\pi') \notin \mathbb{M}$

$M(\pi') \notin L(\mathcal{G})$.



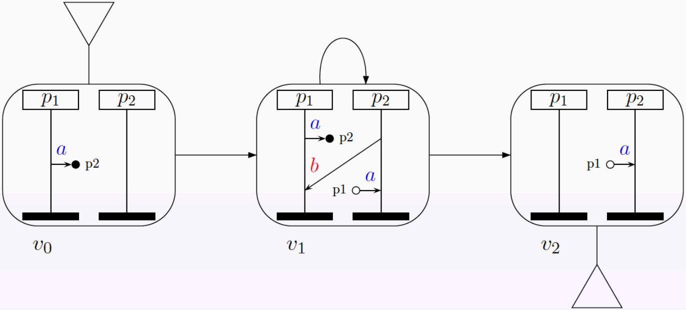
Yannakakis' example as compositional MSG



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

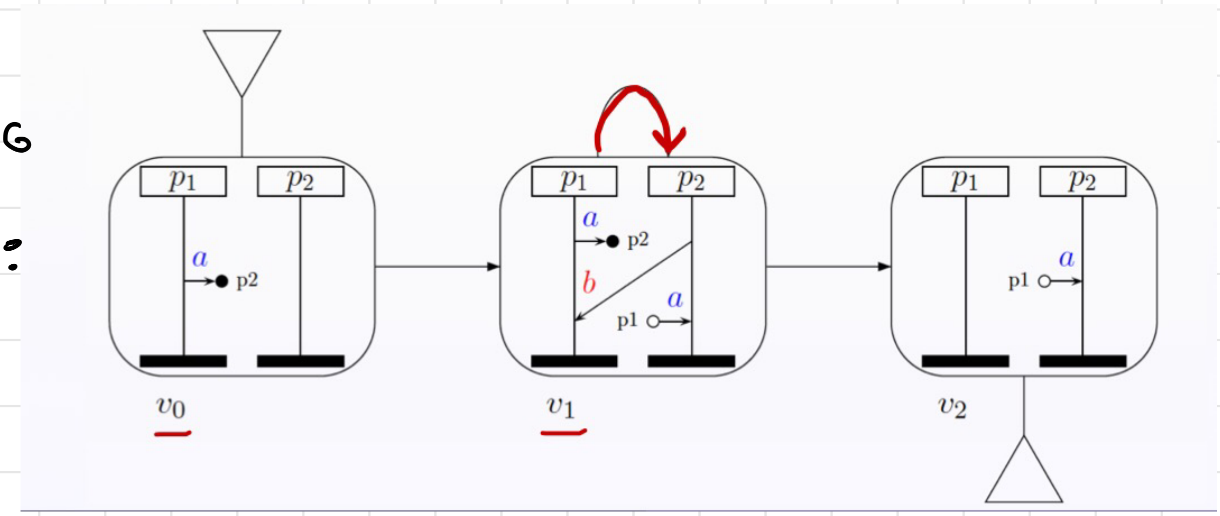
$$M = M_1 \bullet M_2 \bullet \ldots \bullet M_n \quad \text{with} \quad M_i \in \mathbb{M}$$

Thus it cannot be modeled by a MSG.
But it can be modeled as compositional MSG:

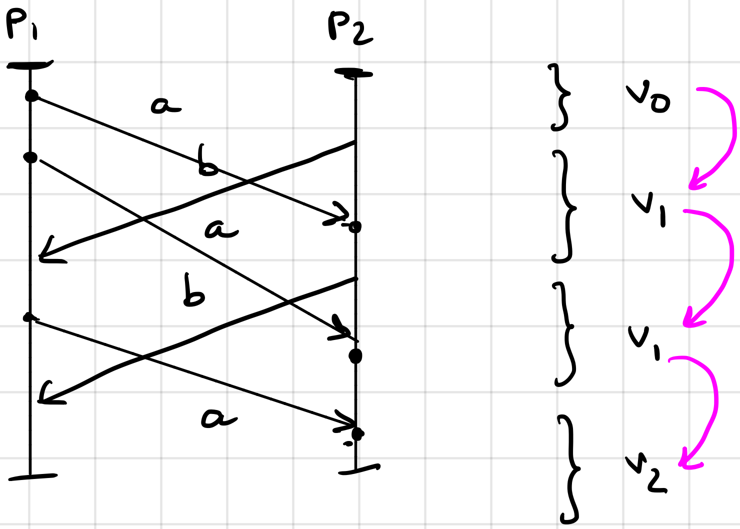


CMSG

$\mathcal{G}$:



MSC $M_1 \in L(\mathcal{G})$



safe

Every accepting path π for G : $\underline{M(\pi) \text{ is an MSC}}$

$\rightarrow M(\pi) \in L(\mathcal{G})$

CMSG $\mathcal{G}$ is called <u>safe</u>

Safe paths and CMSGs

is an MSC

Definition (Safe path)

Path π of CMSG G is safe whenever $\underbrace{M(\pi)}_{\text{CMSC of } \pi} \in \mathbb{M}$.

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 ... $\square$

The complement decision problem "does CMSG G have no safe, accepting path?" is undecidable too.

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.

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.

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 ... □

Pushdown automata

Definition (Pushdown automaton)

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

- Q, a finite set of control states
- $q_0 \in Q$, the initial state
- Γ, a finite stack alphabet — which symbols can be put on the stack
- Σ, a finite input alphabet — a, b, c
- $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q \times \Gamma^*$, the transition relation.

current state

next input symbol that is to be read

top symbol of the stack

next state

next stack content (at the top)

Pushdown automata

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.

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 γ, change to q' and pop γ from the stack.

$L = \{ 0^n 1^n \mid n > 0 \}$

$01 \in L$

$0011 \in L$

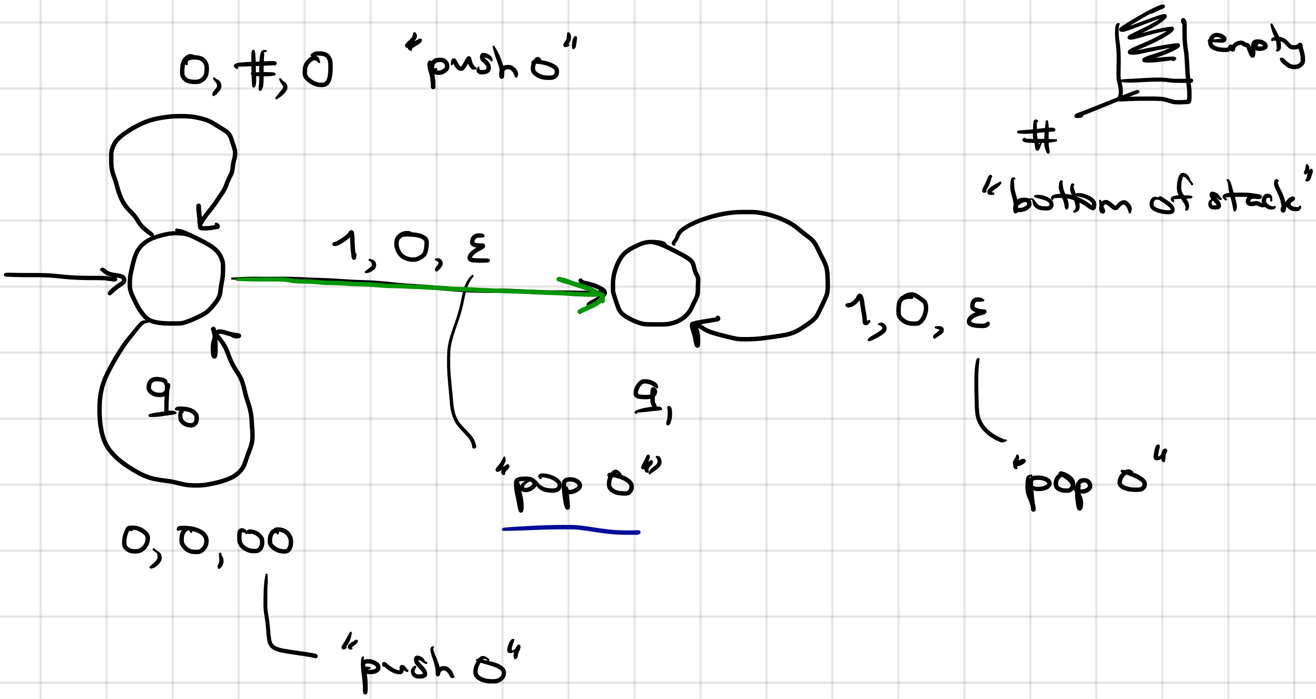
$011 \notin L$

$010 \notin L$

Construct a PDA K such that K accepts the language L

<u>Intuition</u>

- PDA K starts in initial control state q_0
- if input word $w = \varepsilon$ or if w start with a "1" : reject
- otherwise, "scan" all 0s 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 $\underbrace{\text{input word is } \varepsilon \text{ but the stack is not}}_{\text{nr. of 0s} > 1\text{s}}$
- in q_1; accept if input word and the stack are both empty.



$Q = \{q_0, q_1\}$

$\Sigma = \{0, 1\}$

$\Gamma = \{0, \#\}$

$q_0 = q_0$

transitions:

$\underline{(q_0, 1, 0, q_1, \varepsilon)} \in \Delta$

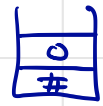
$(q_0, 0, 0, q_0, 00) \in \Delta$

$(q_1, 1, 0, q_1, \varepsilon) \in \Delta$

$(q_0, 0, \#, q_0, 0) \in \Delta$

Example configurations:

$(q_0, \underbrace{0011}_{w}, \underbrace{\#}_{z})$, $(q_1, \underbrace{1}_{w}, \underbrace{0}_{z})$

$z \{$ 

Change of configuration

$(q_0, 0011, \#) \vdash (q_0, 011, 0) \vdash (q_0, 11, 00) \vdash$

$(q_1, 1, 0) \vdash (q_1, \varepsilon, \varepsilon)$

$$\boxed{\text{Is } (q_1, \varepsilon, \varepsilon) \text{ reachable from } (q_0, 0011, \#)\,?}$$

Definition

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

control

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 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: $c_0 \vdash^* c$?

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.

Remark: Dyck language

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

Dyck language

"send" "receive"

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

linearization

Exercise construct a PDA that accepts the Dyck language.

$$(q_0, \underbrace{00111}_{\notin L}, \#) \vdash (q_0, 0011, 0)$$

$$\vdash (q_0, 011, 00)$$

$$\vdash (q_0, 11, 000)$$

$$\vdash (q_1, 1, 00)$$

$$\vdash (q_1, \varepsilon, 0) \qquad \text{"reject"}$$

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:

$$(q_0, 1, \#, q_{err}, \#)$$

$$(q_1, 0, \#, q_{err}, \#)$$

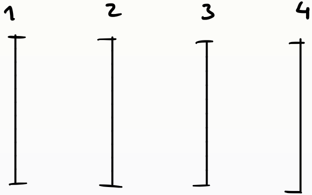
$$(q_1, 0, 0, q_{err}, \#)$$

etcetera.

Checking whether a CMSG is safe is decidable

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

is every accepting path of G safe?



(1,2)	(1,3)	(1,4)
$\boxed{(2,1)}$	(3,1)	(4,1)
(2,3)	(3,2)	$\boxed{(3,4)}$
(2,4)	(4,2)	(4,3)

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

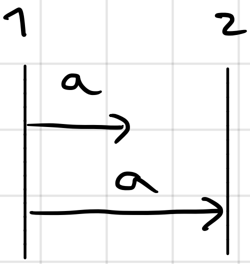
$$\boxed{\text{CMSG } G \text{ is } \underbrace{\text{not safe}}_{\text{in fact "not left-closed"}} \text{ wrt. } (p_i, p_j) \quad \text{iff} \quad \text{PDA } K_{i,j} \text{ accepts}}$$

$K_{i,j}$ is going check for possible violations of being safe

Definition (left-closed CMSC)

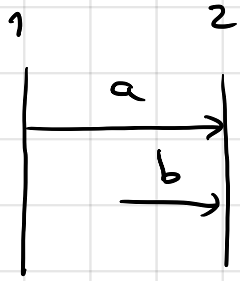
A CMSC is <u>left-closed</u> if it does not contain unmatched receive events, or any send events that are not yet matched and precede other matched send events (of the same type).

Examples



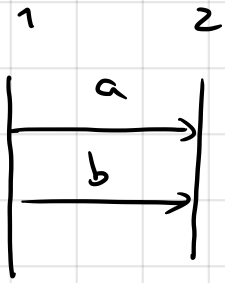
is not left-closed (unsafe)

accept



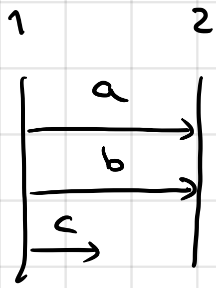
not left-closed (unsafe)

accept



left-closed (and safe)

reject



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

$$\boxed{\text{CMSG } G \text{ is not lc wrt. } (p_i, p_j) \quad \text{iff} \quad \text{PDA } K_{i,j} \text{ accepts}}$$

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

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

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

↳ not a restriction, because such linearisations do always exist

+ assume that matched events are indicated explicitly

!?(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

$$\boxed{\text{CMSG } G \text{ is not lc wrt. } (p_i, p_j) \quad \text{iff} \quad \text{PDA } K_{i,j} \text{ accepts}}$$

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

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

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

a $i \to j$

b at j from 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 $\underbrace{\{a_1, \ldots, a_k\}}$ be the message contents in CMSG G for (p_i, p_j).

all message in CMSG $\mathcal{G}$
send from i to j, or
received at j from i.

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

initial   reject   accept

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

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

  possible symbols in the linearisations
- Input alphabet

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

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 Δ is described on next slide

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

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

"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

pending unmatched send $p_i \to p_j$

- On reading $?(p_j, p_i, a_m)$ in $\underline{q_0}$, proceed as follows:
  - if 1 is on stack, pop it
  - otherwise, i.e., if stack is empty, accept (i.e., move to q_F)

not 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

not 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
- 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)

left-closed.

X
- 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

still pending sends/receives

- Remaining input w empty? Accept, if stack non-empty; else reject

Safeness of CMSGs (3)

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

- Ignore all actions except $?(p_j, p_i, a_\ell)$ for all $0 < \ell \leqslant k$

Safeness of CMSGs (3)

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

- Ignore all actions except $?(p_j, p_i, a_\ell)$ for all $0 < \ell \leqslant k$
- On reading $?(p_j, p_i, a_\ell)$ (for some $0 < \ell \leqslant k$) in state q_{a_m} do:
  - if 1 is on top of stack, pop it

Safeness of CMSGs (3)

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

- Ignore all actions except $?(p_j, p_i, a_\ell)$ for all $0 < \ell \leqslant k$
- On reading $?(p_j, p_i, a_\ell)$ (for some $0 < \ell \leqslant k$) in state q_{a_m} do:
  - if 1 is on top of stack, pop it

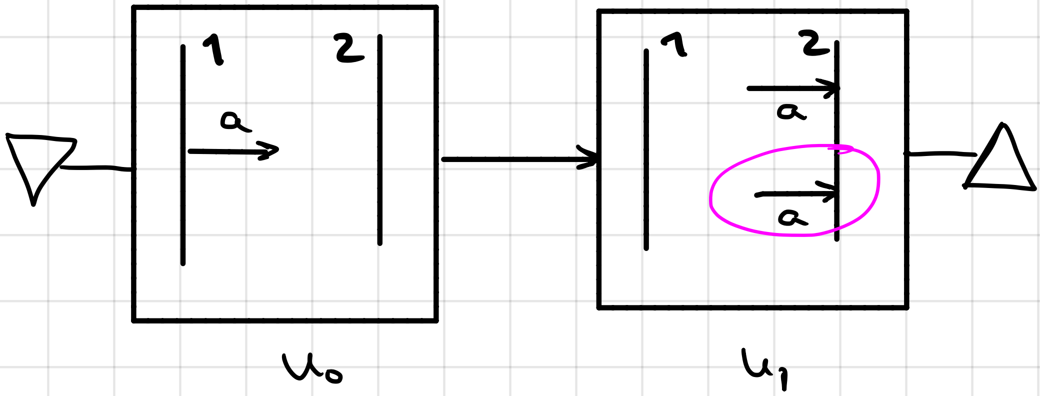
not left-closed

- If stack is empty:
  - if last receive differs from a_m, accept
  - otherwise reject, while ignoring the rest (if any) of the input

left-closed

Example 1.

CMSG $\mathcal{G}_1$



$$\underbrace{(q_0, \#, !a?a?a)}_{\text{initial configuration}} \vdash (q_a, 1, ?a?a) \vdash (q_a, \#, ?a)$$

reject

$$(q_0, \#, !a?a?a) \vdash (q_0, 1, ?a?a) \vdash (q_0, \#, ?a)$$

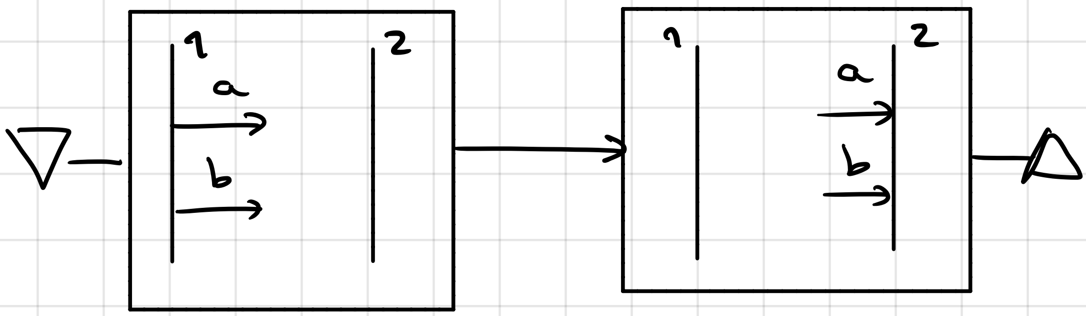
accept

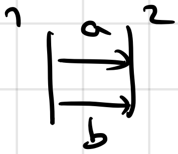
Thus the PDA $K_{1,2}$ accepts the input word $\Longrightarrow$ $\mathcal{G}_1$ is not left-closed

Example 2

CMSG $\mathcal{G}_2$:

safe.



$L(\mathcal{G}) =$ 

PDA $\mathcal{K}_{1,2}$

$(q_0, \#, !a!b?a?b) \vdash (q_a, 1, !b?a?b)$

$(q_0, \#, !a!b?a?b) \vdash (q_0, 1, !b?a?b)$

$(q_0, 1, !b?a?b) \vdash (q_0, 11, ?a?b) \vdash (q_0, 1, ?b) \vdash (q_0, \#, \varepsilon)$ reject

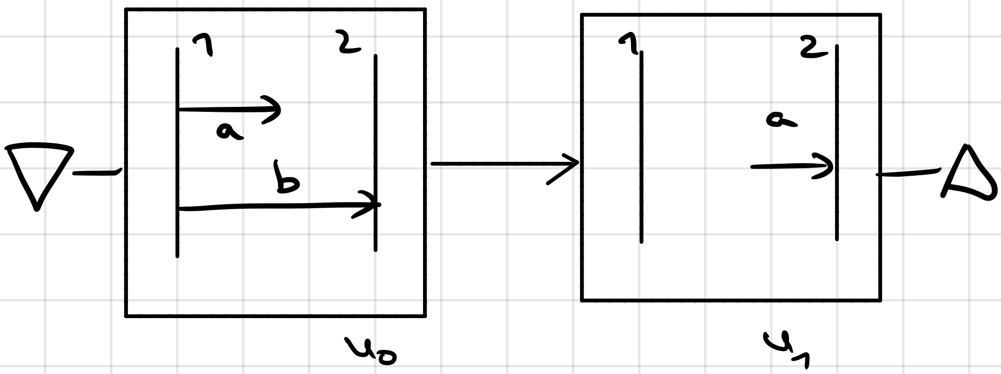
$(q_0, 1, !b?a?b) \vdash (q_b, 11, ?a?b) \vdash (q_b, 1, ?b) \vdash (q_b, \#, \varepsilon)$ reject

$(q_a, 1, !b?a?b) \vdash (q_a, 1, ?a?b) \vdash (q_a, \#, ?b) \vdash (q_a, \#, \varepsilon)$ reject

PDA $\mathcal{K}_{1,2}$ has no accepting run on $!a!b?a?b$ and thus CMSG $\mathcal{G}_2$ is <u>left-closed</u>

Example

CMSG $\mathcal{G}_3$



PDA $K_{1,2}$

$(q_0, \#, \ !a\ !?b\ ?a) \vdash (q_0, 1, \ !?b\ ?a)$

accept

$\top$

$(q_a, 1, \ !?b\ ?a)$

$\longrightarrow$ violation of FIFO property.

$\top$

$(q_a, 1, ?a)$

$\top$

$(q_a, \#, \varepsilon)$

reject

PDA $K_{1,2}$ accepts

$\Longrightarrow$ CMSG $\mathcal{G}_3$ is not left-closed

$\Longrightarrow$ is not safe.

Safeness of CMSGs (4)

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

$\Longrightarrow$ CMSG G is not lc wrt. (p_i, p_j) iff configuration $\underbrace{(q_F, \cdot, \cdot)}_{\text{accept}}$ is reachable.

Safeness of CMSGs (4)

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

$\Longrightarrow$ CMSG G is not lc wrt. (p_i, p_j) iff configuration $(q_F, \cdot, \cdot)$ is reachable.

$\Longrightarrow$ reachability of a configuration in a PDA is in PTIME, hence checking safeness wrt. (p_i, p_j) is in PTIME.

Esparza et al.

Safeness of CMSGs (4)

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

$\Longrightarrow$ CMSG G is not lc wrt. (p_i, p_j) iff configuration $(q_F, \cdot, \cdot)$ is reachable.

$\Longrightarrow$ 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 $\mathcal{O}(k^2 \cdot N^2 \cdot L \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $L = |\mathcal{C}|$.

lc

Safeness of CMSGs (4)

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

$\Longrightarrow$ CMSG G is not safe wrt. (p_i, p_j) iff configuration $(q_F, \cdot, \cdot)$ is reachable.

$\Longrightarrow$ 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 $\mathcal{O}(k^2 \cdot N^2 \cdot L \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $L = |\mathcal{C}|$.

Lc

Proof.

Checking reachability in PDA $K_{i,j}$ is in $\mathcal{O}(L \cdot |E|^2)$.

Safeness of CMSGs (4)

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

$\Longrightarrow$ CMSG G is not safe wrt. (p_i, p_j) iff configuration $(q_F, \cdot, \cdot)$ is reachable.

$\Longrightarrow$ 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 $\underbrace{\text{safe}}_{k}$ is in $\mathcal{O}(k^2 \cdot N^2 \cdot L \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $L = |\mathcal{C}|$.

Proof.

Checking reachability in PDA $K_{i,j}$ is in $\mathcal{O}(L \cdot |E|^2)$. 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)

$\Longrightarrow$ CMSG G is not safe wrt. (p_i, p_j) iff configuration $(q_F, \cdot, \cdot)$ is reachable.

$\Longrightarrow$ 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 $\underbrace{\text{safe}}_{\text{lc}}$ is in $\mathcal{O}(k^2 \cdot N^2 \cdot L \cdot |E|^2)$ where $k = |\mathcal{P}|$, $N = |V|$, and $L = |\mathcal{C}|$.

Proof.

Checking reachability in PDA $K_{i,j}$ is in $\mathcal{O}(L \cdot |E|^2)$. 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 $\mathcal{O}(N^2)$, as a single traversal for each loop in G suffices. □