FOUNDATIONS OF THE UML

Lecture 3:

Compositional MSGs

Joost-Pieter Katoen

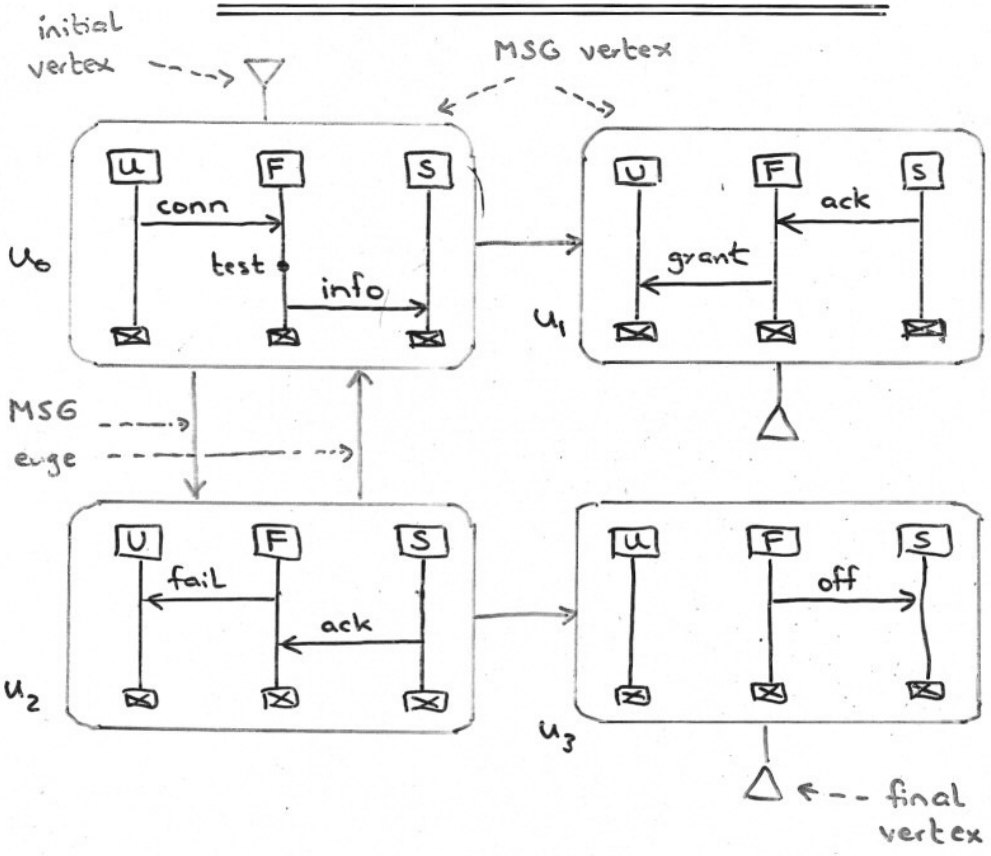
MOVES @ RWTH

moves.rwth-aachen.de /i2/1g2

WS 2007/2008

November 14, 2007

MESSAGE SEQUENCE GRAPHS



$u_0\ u_2\ u_0\ u_1 = \lambda(u_0) \cdot \lambda(u_2) \cdot \lambda(u_0) \cdot \lambda(u_1)$

DEFINITION

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

A Message sequence Graph (MSG) G is a tuple $G = (V, \longrightarrow, v_0, F, \lambda)$ with:

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

Note: 1. an MSG is an NFA without input alphabet where states are MSCs

2. every MSC is an MSG

3. generalization towards a set of initial vertices is straightforward

CONCATENATION OF MSCs

Let $M_i = (P_i, E_i, \mathbb{C}_i, \ell_i, m_i, <_i)$ $\quad i \in \{1,2\}$ be an MSC with $E_1 \cap E_2 = \emptyset$

The concatenation of M_1 and M_2 is the MSC

$$M_1 \cdot M_2 = (P, E, \mathbb{C}, \ell, m, <) \quad \text{with:}$$

$$P = P_1 \cup P_2 \qquad E = E_1 \cup E_2 \qquad \mathbb{C} = \mathbb{C}_1 \cup \mathbb{C}_2$$

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

$$\ell(e) = \begin{cases} \ell_1(e) & \text{if } e \in E_1 \\ \ell_2(e) & \text{if } e \in E_2 \end{cases} \qquad m(e) = \begin{cases} m_1(e) & \text{if } e \in E_1 \\ m_2(e) & \text{if } e \in E_2 \end{cases}$$

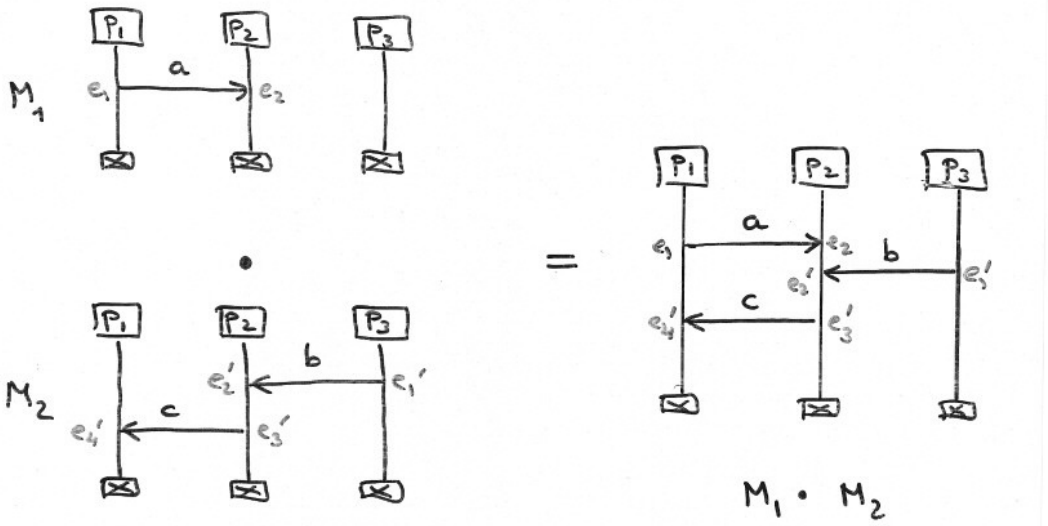
$$< = <_1 \cup <_2 \cup \{ (e,e') \mid e \in E_1 \cap E_p,\ e' \in E_2 \cap E_p \}$$

(same process)

Note:

- events are ordered process-wise
  events at p in M_1 precede events at p in M_2
- thus: some processes may proceed to M_2 before others!
- $\neq$: first complete M_1 then execute M_2 → see Exercises

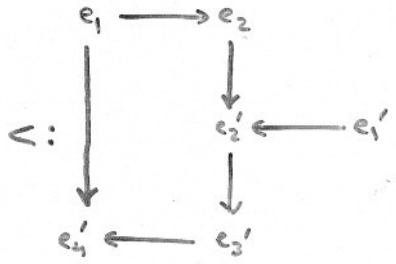
EXAMPLE



$<_1$: $e_1 \longrightarrow e_2$

$e_2' \longleftarrow e_1'$

$<_2$: $e_4' \longleftarrow e_3'$



Note: e_1 and e_1' are not ordered in $M_1 \cdot M_2$

e.g. $e_1\, e_2\, e_1'\, e_2' \ldots \in Lin(M_1 \cdot M_2)$

$e_1'\, e_1\, e_2\, e_2' \ldots \in Lin(M_1 \cdot M_2)$

LANGUAGE OF AN MSG

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

A path π of G is a finite sequence

$\pi = u_0 u_1 \ldots u_n$ with $u_i \in V$ $\quad 0 \le i \le n$

$u_i \rightarrow u_{i+1}$ $\quad 0 \le i < n$

The MSC of a path $\pi = u_0 \ldots u_n$ is:

$$M(\pi) = \lambda(u_0) \cdot \lambda(u_1) \cdot \ldots \cdot \lambda(u_n) = \prod_{i=0}^{n} \lambda(u_i)$$

(MSC concatenation)

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

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

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

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

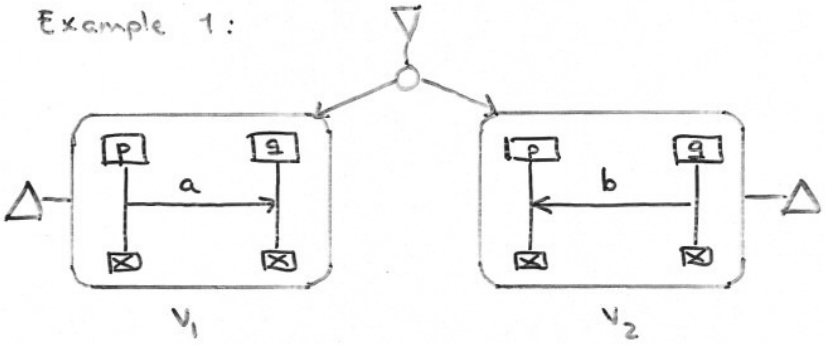
where $Lin(\{M_1, \ldots, M_k\}) = \bigcup_{i=1}^{k} Lin(M_i)$

EXPRESSIVENESS

- The decision problem "MSG G has a race" is undecidable
- MSGs may represent infinite-state systems
- State space of an MSG may not be CF
- State space of an MSG is context-sensitive
- The decision problem: for MSGs G_1 and G_2, do we have $L(G_1) \cap L(G_2) = \phi$ is undecidable
- Corollary: the decision problem for MSG G and DFA A, do we have $Lin(L(G)) \cap L(A) = \phi$ is undecidable

LOCAL CHOICE PROPERTY

Example 1:



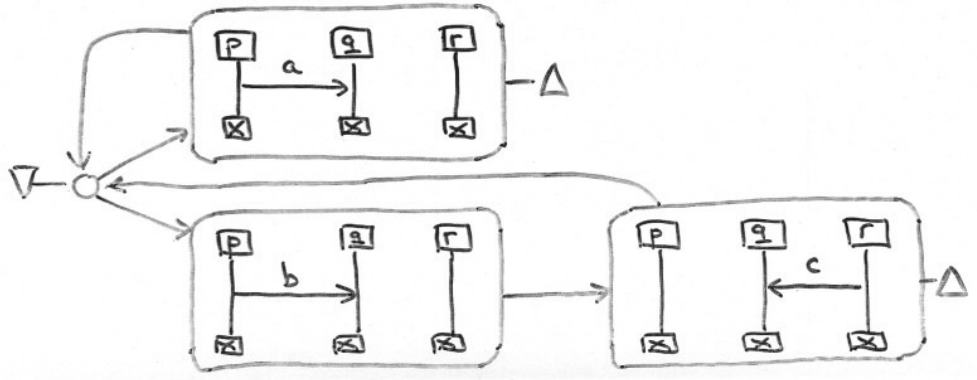
Inconsistency if p behaves according to V_1 and q behaves according to V_2

$\Rightarrow$ possible distributed realization may yield a deadlock

(this will be made more precise in next lecture)

Problem: subsequent behaviour is determined by distinct processes

Example 2:



LOCAL CHOICE PROPERTY

- e is a minimal event wrt. $<$ if $\neg(\exists e'.\, e'<e)$
- p is active in MSC M if $E_p \neq \emptyset$

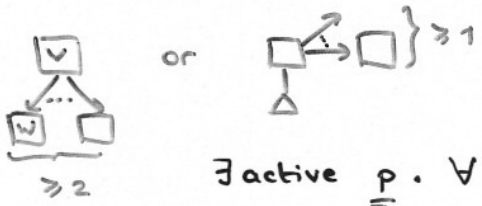
  p is active in $v_1 v_2 \ldots v_n$ in MSG G if

  $\exists v_i .$ p is active in $\lambda(v_i)$ $\quad 1 \leq i \leq n$

- Def: MSG $G = (V, \rightarrow, v_0, F, \lambda)$ is <u>local choice</u> if:

  (1) $\exists$ active p. $\forall \pi \in$ Paths (v_0).
  π has a single minimal event e with $e \in E_p$

  (2) $\forall$ <u>branching vertex</u> $v \in V$.

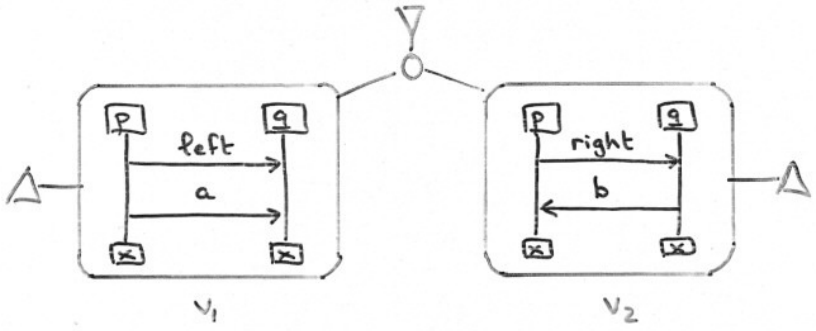


$\exists$ active <u>p</u> . $\forall \pi \in$ Paths (w) . and $v \rightarrow w$

π has a single minimal event e with $e \in E_p$

Intuition: along every path from an initial or branching vertex there is a <u>single</u> process deciding how to proceed which can inform the other processes how to proceed

LOCAL CHOICE



is local choice

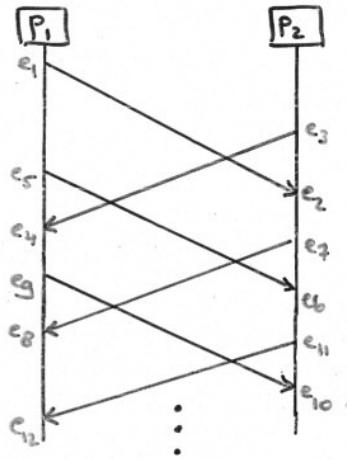
Checking whether an MSG is local choice can be done in polynomial time

How can non-local choices be resolved?

Refine your MSG and add control messages (cf. above example)

RESTRICTION OF MSGs

[Yannakakis]



This MSC cannot be decomposed as $M_1 \circ M_2 \circ \ldots \circ M_n \quad (n>1)$

This can be seen as follows:

- e_1 and $e_2 = m(e_1)$ must reside in same M_i
- $e_3 < e_2$ and $e_1 < e_4$ thus $e_3, e_4 \notin M_j$, $j<i$ or $j>i$ $\Rightarrow$ $e_3, e_4 \in M_i$
- by similar reasoning: $e_5, e_6 \in M_i$ etc.

Problem: compulsory matching between send and receive in same MSG vertex

(i.e. send e and receive m(e))

COMPOSITIONAL MSCs

[Gunter, Muscholl, Peled 2001]

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

(= allow for incomplete message transfer)

Def $M = (P, E, \mathbb{C}, \ell, m, <)$ is a compositional MSC where $P, E, \mathbb{C}$ and ℓ are as before, and

- $m: E_! \to E_?$ is a partial, injective function (not a bijection!) such that (as before):

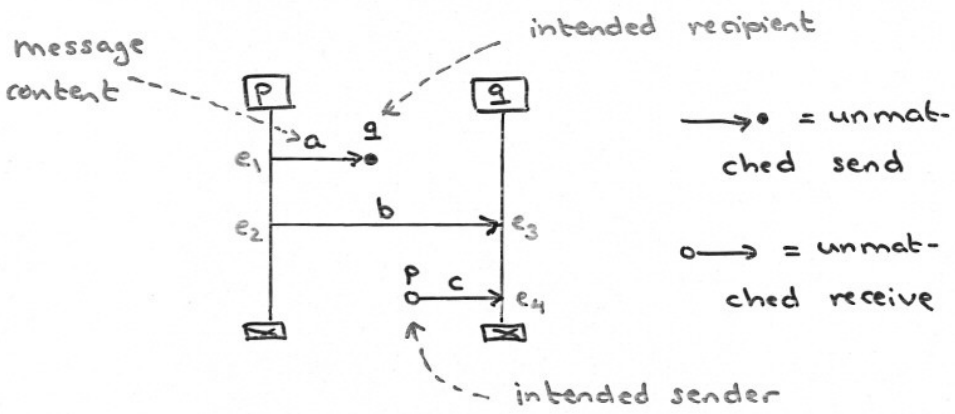
  $m(e) = e' \wedge \ell(e) = p!q(a) \longrightarrow \ell(e') = q?p(a)$

- $< = \bigcup_{p \in P} <_p \;\cup\; \{(e, m(e)) \mid e \in \underbrace{dom(m)}_{\text{domain of } m}\}$

  "m(e) is defined"

Note: an MSC is a CMSC where m is total and bijective.

CMSC EXAMPLE



$m(e_2) = e_3$

$e_1 \notin dom(m)$

$e_4 \notin rng(m)$

Def A compositional MSG (CMSG)

$G = (V, \rightarrow, v_0, F, \lambda)$ with $\lambda : V \rightarrow \mathbb{CM}$

and $V, \rightarrow, v_0$ and F as before

$\mathbb{CM}$: set of CMSCs

CONCATENATION OF CMSCs

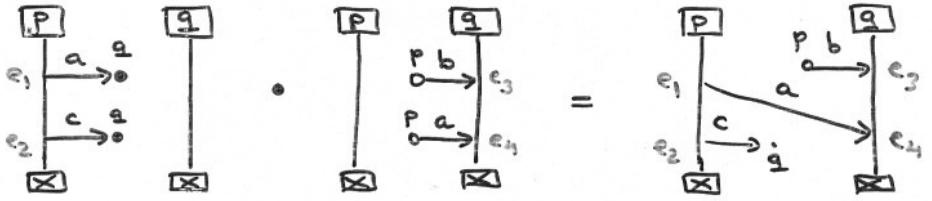
Let $M_i = (P_i, E_i, \mathbb{C}_i, \ell_i, m_i, <_i) \in \mathbb{CM}$, $i \in \{1,2\}$

$E_1 \cap E_2 = \emptyset$

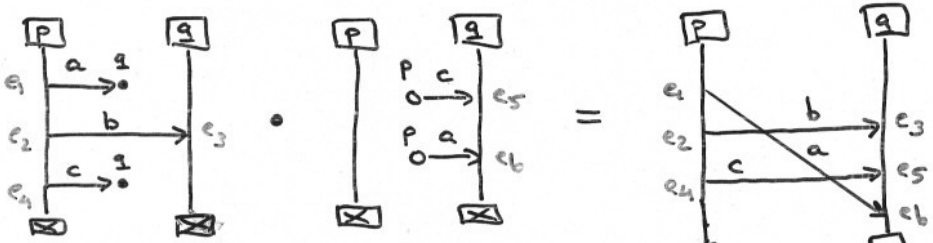
Then: $M_1 \cdot M_2 = (P_1 \cup P_2, E_1 \cup E_2, \mathbb{C}_1 \cup \mathbb{C}_2, \ell, m, <)$ with:

- $\ell(e) = \ell_1(e)$ if $e \in E_1$, $\ell(e) = \ell_2(e)$ otherwise
- $m: E_! \rightarrow E_?$ satisfies:
  1) m extends m_1 and m_2
     $e \in dom(m_i)$ 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 processes
     (= matching from top-to-bottom)
     the k-th unmatched send in M_1 is matched with k-th unmatched receive in M_2 (of the same "type")
  3) $M_1 \cdot M_2$ satisfies the FIFO condition (when restricted to matched events)
- $< = \left(\bigcup_{p \in P} <_{p,1} \cup <_{p,2} \right) \cup \{ (e,e') \mid e \in E_1 \cap E_p, e' \in E_2 \cap E_p \}$
  $\cup \{ (e, m(e)) \mid e \in dom(m) \}$

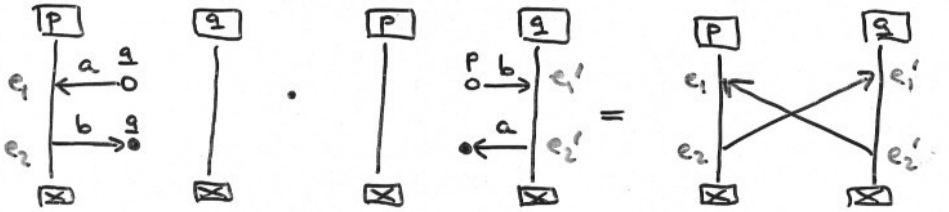
EXAMPLES



M_1 M_2



M_1 M_2 non-FIFO!

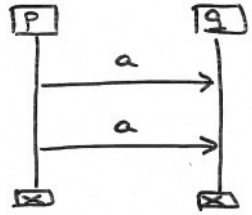


M_1 M_2 cyclic!

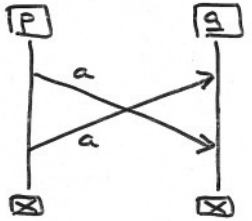
ASSOCIATIVITY

M M'

$(M \cdot M) \cdot M'$:



$M \cdot (M \cdot M')$:



this is not FIFO

(and thus undefined)

So: concatenation of CMSGs is <u>not</u> associative.

LANGUAGE OF A CMSG

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

A path π of G is a finite sequence

$\pi = u_0 u_1 \ldots u_n$ with $u_i \in V$, $u_i \longrightarrow u_{i+1}$

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

$$M(\pi) = \left(\ldots (\lambda(u_1) \cdot \lambda(u_2)) \cdot \lambda(u_3) \ldots \right) \cdot \lambda(u_n)$$

$$= \prod_{i=0}^{n} \lambda(u_i)$$

CMSC concatenation which is left-associative

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

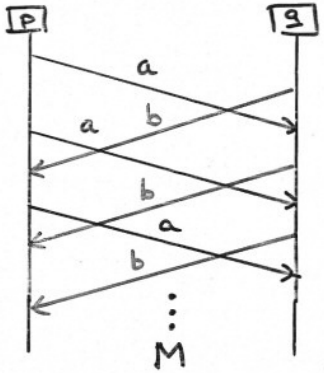
The (MSC) language of CMSG G is defined by:

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

note: only MSCs are considered

G is safe if for every accepting path π of G, $M(\pi)$ is an MSC (no unmatched sends / receipts)

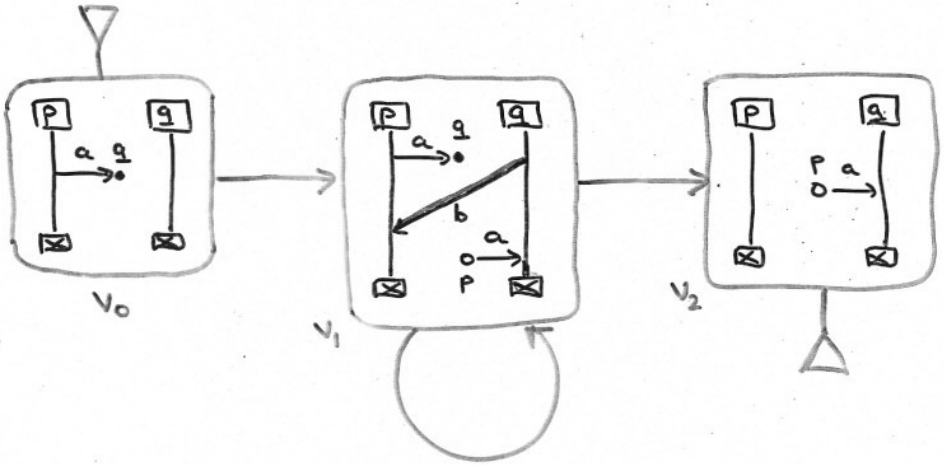
CONSIDER AGAIN



recall: this could not be modeled as

$M = M_1 \cdot M_2 \ldots \cdot M_n$ $(n > 1)$

M_i is an MSC



The CMSG for the above MSC

SOME PROPERTIES OF CSMGs

- CMSGs are strictly more expressive than MSGs
- There are subclasses of CMSGs that are equally expressive to MSGs
  so-called locally safe CSMGs
- Let path π of CSMG G such that $M(\pi) \in \mathbb{M}$
  Such paths are called safe.
- The decision problem "does G have at least one safe, accepting path?" is undecidable
- The decision problem "is CSMG safe?" is decidable in PTIME

EXISTENCE OF SAFE PATHS

The decision problem:

INPUT a CSMG G

OUTPUT YES, if G has a safe, accepting path

NO, otherwise

is undecidable

Proof idea: reduction from the Post's Correspondence Problem (PCP).

- Let PCP (U,W) over alphabet Σ
  with $U = u_1, u_2, \ldots, u_m$
  $W = w_1, w_2, \ldots, w_m$
  $u_i, w_i \in \Sigma^*$
  i.e., instance $\langle (u_1,w_1), \ldots, (u_m,w_m) \rangle$
- Construct a CMSG $G_{U,W}$ such that

(*) PCP (U,W) has a solution iff CSMG G has an accepting, safe path

- PCP is undecidable and (*) imply:
  the above decision problem is undecidable

SAFENESS OF CMSGs

The decision problem "is CMSG G safe?" is decidable in PTIME

- Consider a pair of processes (p_i, p_j)
- Construct a push-down automaton

$K_{i,j} = (Q, \Gamma, \Sigma, \Delta)$ with :

- Q a finite set of control states
- Γ = stack alphabet = $\{\uparrow, \bot\}$ (↑ counter, $\bot$ stack bottom)
- $\Sigma = \begin{cases} \text{unmatched } !(a) \\ \text{unmatched } ?(a) \\ \text{matched } (a) \end{cases}$ for $a \in \mathbb{C}$
- $\Delta \subseteq (Q \times \Sigma \times \Gamma) \times (Q \times \{\text{push}, \text{pop}, \text{skip}\})$

$(q, a, \uparrow, q', \text{pop})$ means : on reading a and top stack is $\uparrow$ in state q, change to q' and pop $\uparrow$

- accept if p_i and p_j completed, and stack is nonempty or if unmatched $p_i!p_j(a)$ is matched by $p_j?p_i(b)$ with $a \neq b$

SAFENESS OF CMSGs

Functioning of PDA $K_{i,j}$:

Phase 1 : replace vertex v in G by a linearization of $\lambda(v)$ s.t. all unmatched receive events precede all unmatched send events of same type

$K_{i,j}$ follows events in $\lambda(v)$ and continues nondeterministically to $\lambda(w)$ for some direct successor w of v in G

Phase 2 : push $\uparrow$ on stack if $p_i!p_j(a)$ is unmatched, pop $\uparrow$ if $p_j?p_i(a)$ is not matched. On occurrence of $p_i!p_j(a)$ move to control state $q_a \in Q$, ignore all events except unmatched receipts (pop!); stack empty? compare message content of last receive : if $a \neq b \longrightarrow$ accept ;

otherwise : ignore rest of events + continue

SAFENESS OF CSMGs

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

$\Rightarrow$ checking safeness amounts to checking reachability in all PDAs $K_{i,j}$.

$\Rightarrow$ reachability of a configuration in a PDA is decidable in PTIME, hence checking safeness can be done in PTIME

Note: in above construction we assumed (for simplicity) that each safe path is also well-formed (see next slide).

Without this assumption, the PDA $K_{i,j}$ needs to be adapted (how?)

WELL-FORMED PATHS

Let CSMC M be <u>well-formed</u> if:

- $E_? = \mathrm{rng}(m)$
- for all $e, e' \in E$:

$$\ell(e) = p!g(a) \wedge e \notin \mathrm{dom}(m) \wedge \ell(e') = \ell(e) \wedge e' \in \mathrm{dom}(m)$$

implies $e' <_p e$

Path $v_1 \dots v_k$ in CMSG G is <u>well-formed</u> if for all $n \leq k$, $\prod_{i=1}^{n} \lambda(v_i)$ is a well-formed CMSC.