Realizability of Dynamic MSC Languages

Benedikt Bollig[1] and Loïc Hélouët[2]

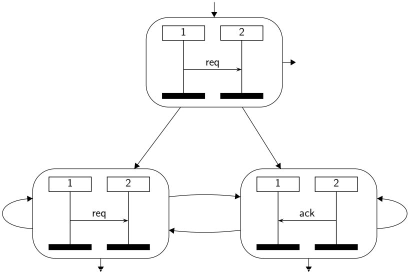
[1]LSV, ENS Cachan, CNRS

[2]IRISA, INRIA, Rennes

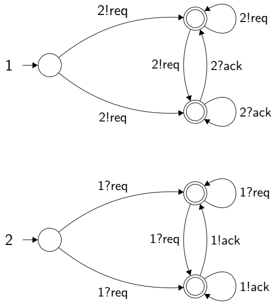
Automata, Concurrency and Timed Systems (ACTS) II

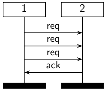
Chennai Mathematical Institute
February 1–3, 2010

Realizability of Message Sequence Charts



$\stackrel{?}{\Longrightarrow}$





[AEY'05] Alur & Etassami & Yannakakis. Realizability and Verification of Message Sequence Graphs. 2001/2005.

[L'03] Lohrey. Realizability of high-level message sequence charts: closing the gaps. 2003.

[HMNST'05] Henriksen & Mukund & Narayan Kumar & Sohoni & Thiagarajan. A Theory of Regular MSC Languages. 2005.

Specification formalisms for distributed systems

Specification

finite automata
high-level MSCs
monadic second-order logic
dynamic MSC grammars
...

Synthesis →

Implementation

asynchronous automata
communicating automata
dynamic communicating automata
...

Specification formalisms for distributed systems

Specification

finite automata
high-level MSCs
monadic second-order logic
dynamic MSC grammars
...

Synthesis →

Implementation

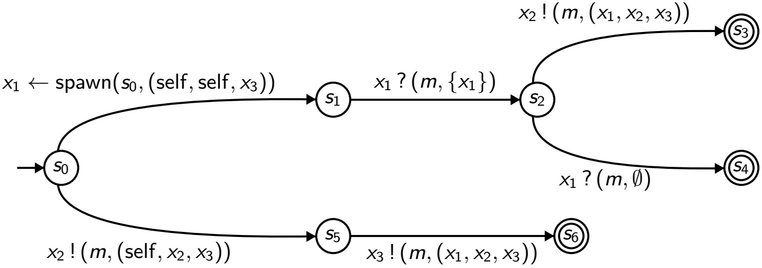
asynchronous automata
communicating automata
dynamic communicating automata
...

1. Dynamic Communicating Automata
2. Dynamic MSC Grammars
3. Realizability
4. Implementation

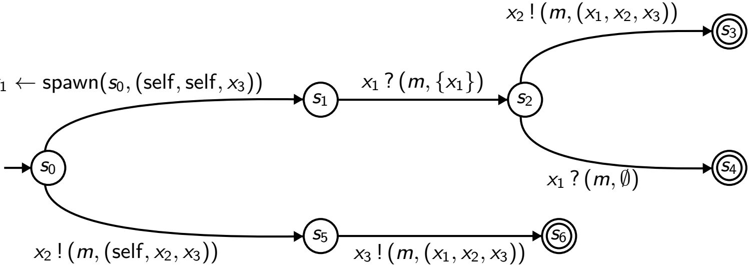
Presentation outline

1. Dynamic Communicating Automata
2. Dynamic MSC Grammars
3. Realizability
4. Implementation

Dynamic Communicating Automaton

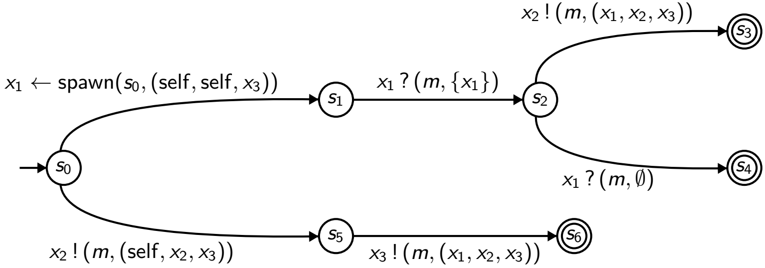


Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3
1	s_0	1	1	1

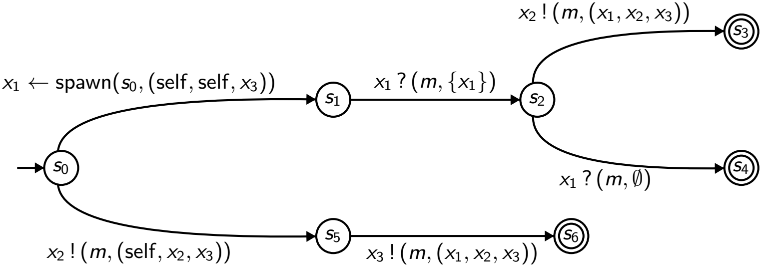
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2
1	s_1	2	1	1	—	
2	s_0	1	1	1		—

spawn(1, 2)

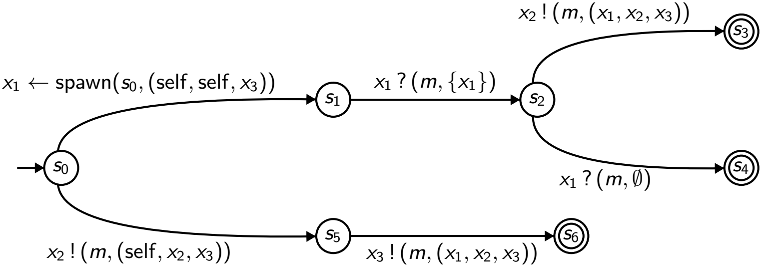
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3
1	s_1	2	1	1	—		
2	s_1	3	1	1		—	
3	s_0	2	2	1			—

spawn(1, 2) spawn(2, 3)

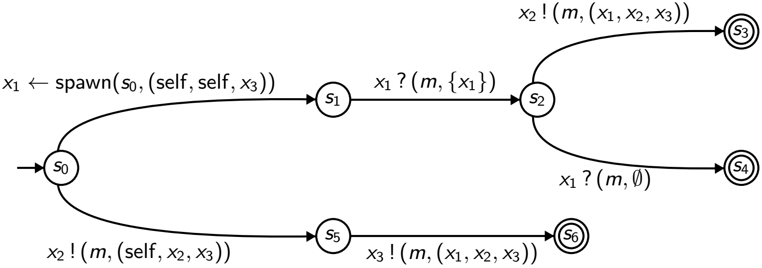
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_1	2	1	1	—			
2	s_1	3	1	1		—		
3	s_1	4	2	1			—	
4	s_0	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4)

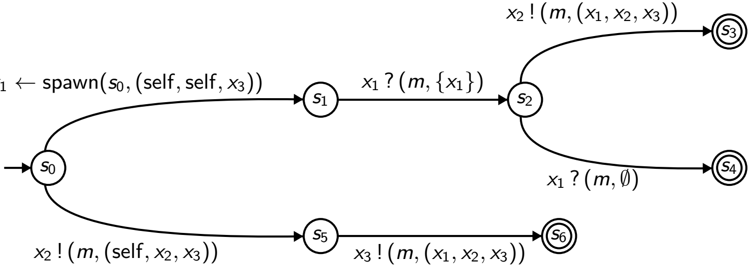
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_1	2	1	1	—			
2	s_1	3	1	1		—		
3	s_1	4	2	1			—	$(4, 3, 1)$
4	s_5	3	3	1				—

$\text{spawn}(1, 2)\ \text{spawn}(2, 3)\ \text{spawn}(3, 4)\ !(4, 3)$

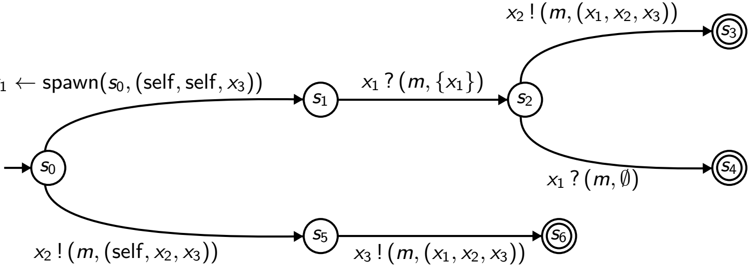
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_1	2	1	1	—			(3, 3, 1)
2	s_1	3	1	1		—		
3	s_1	4	2	1			—	(4, 3, 1)
4	s_6	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1)

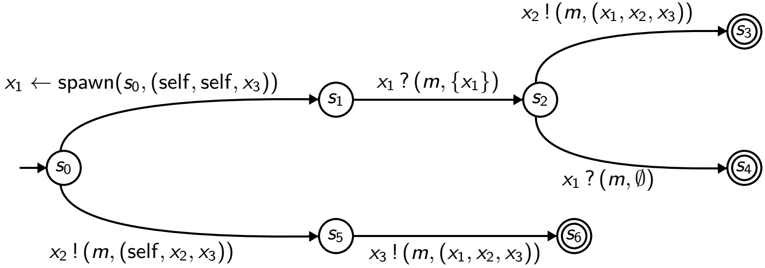
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_1	2	1	1	—			(3, 3, 1)
2	s_1	3	1	1		—		
3	s_2	4	2	1			—	
4	s_6	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3)

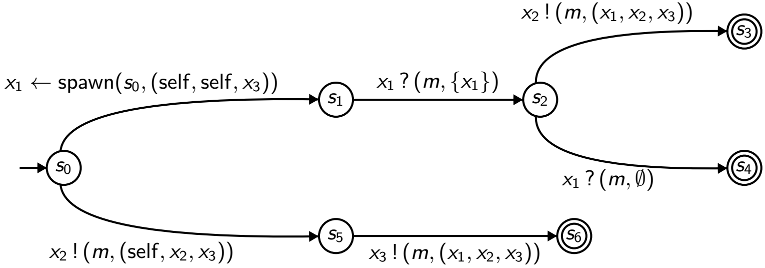
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_1	2	1	1	—			$(3, 3, 1)$
2	s_1	3	1	1		—	$(4, 2, 1)$	
3	s_3	4	2	1			—	
4	s_6	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3) !(3, 2)

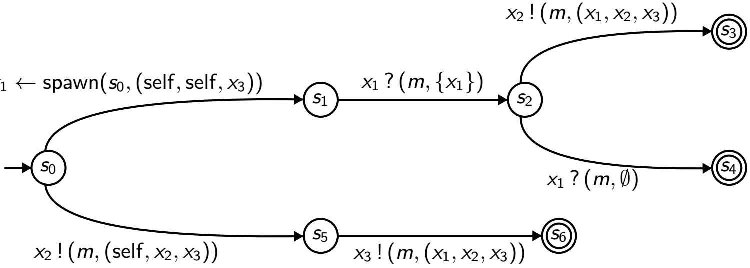
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_1	2	1	1	—			(3, 3, 1)
2	s_2	4	1	1		—		
3	s_3	4	2	1			—	
4	s_6	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3) !(3, 2) ?(3, 2)

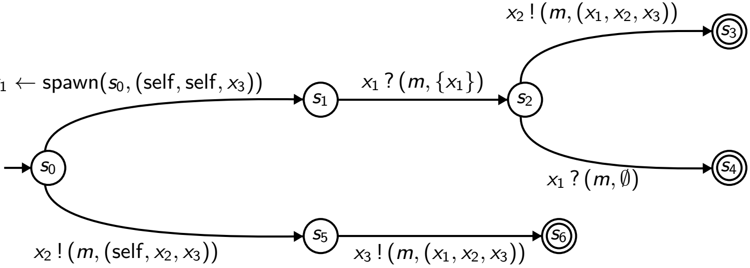
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_1	2	1	1	—	$(4, 1, 1)$		$(3, 3, 1)$
2	s_3	4	1	1		—		
3	s_3	4	1	1			—	
4	s_6	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3) !(3, 2) ?(3, 2) !(2, 1)

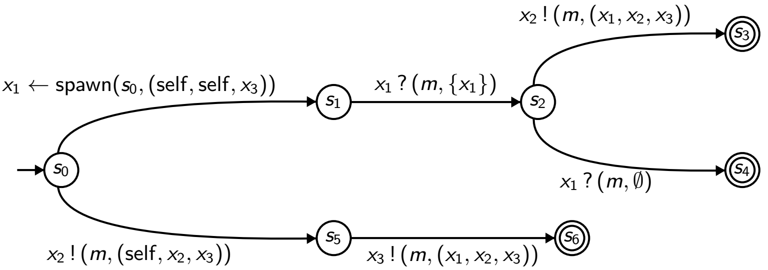
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_2	4	1	1	—			(3, 3, 1)
2	s_3	4	1	1		—		
3	s_3	4	1	1			—	
4	s_6	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3) !(3, 2) ?(3, 2) !(2, 1) ?(2, 1)

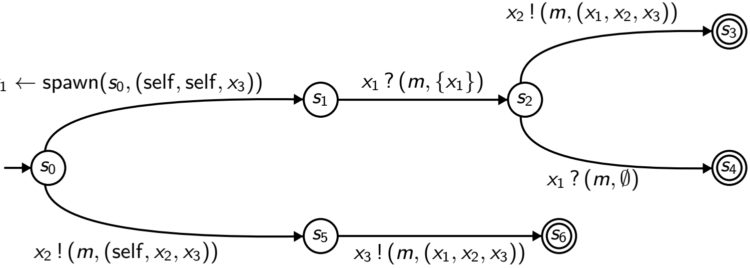
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_4	4	1	1	—			
2	s_3	4	1	1		—		
3	s_3	4	1	1			—	
4	s_6	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3) !(3, 2) ?(3, 2) !(2, 1) ?(2, 1) ?(1, 4)

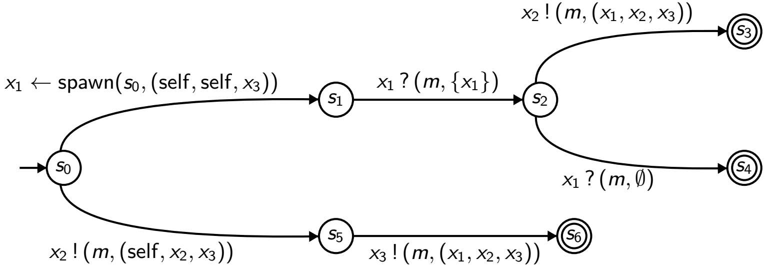
Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
1	s_4	4	1	1	—			
2	s_3	4	1	1		—		
3	s_3	4	1	1			—	
4	s_6	3	3	1				—

spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3) !(3, 2) ?(3, 2) !(2, 1) ?(2, 1) ?(1, 4)

Dynamic Communicating Automaton



Proc	State	x_1	x_2	x_3	1	2	3	4
9	s_4	4	9	9	—			
7	s_3	4	9	9		—		
2	s_3	4	7	9			—	
4	s_6	2	2	1				—

spawn(9, 7) spawn(7, 2) spawn(2, 4) !(4, 2) !(4, 9) ?(4, 2) !(2, 7) ?(2, 7) !(7, 9) ?(7, 9) ?(9, 4)

Dynamic Communicating Automaton

Definition

A DCA is a tuple $\mathcal{A} = (X, Msg, Q, \Delta, \iota, F)$ where

- X set of process variables
- Msg set of messages
- Q set of states
- $\iota \in Q$ initial state
- $F \subseteq Q$ set of final states
- $\Delta \subseteq Q \times Act_{\mathcal{A}} \times Q$ set of transitions

Dynamic Communicating Automaton

Definition

The set $Act_{\mathcal{A}}$ of actions contains

- $x \leftarrow \text{spawn}(s, \eta)$ spawn action
- $x\,!\,(m, \eta)$ send action
- $x\,?\,(m, Y)$ receive action
- $\text{rn}(\sigma)$ variable renaming

for all

$$\begin{array}{lll} x \in X & s \in Q & \eta : \; X \to (X \uplus \{\text{self}\}) \\ Y \subseteq X & m \in Msg & \sigma : \; X \to X \end{array}$$

Dynamic Communicating Automaton

Configuration

Quadruple $(\mathcal{P}, state, proc, ch)$ where

- $\mathcal{P} \subseteq \mathbb{N}$ nonempty finite
- $state : \mathcal{P} \to Q$
- $proc : \mathcal{P} \to \mathcal{P}^X$
- $ch : (\mathcal{P} \times \mathcal{P}) \to \left(Msg \times \mathcal{P}^X\right)^*$

Transition system

- initial configuration: $(\{p\}, p \mapsto \iota, (p, x) \mapsto p, (p, p) \mapsto \epsilon)$
- final configuration: all states final and all channels empty
- $\Longrightarrow_{\mathcal{A}} \subseteq Conf_{\mathcal{A}} \times (\Sigma \cup \{\epsilon\}) \times Conf_{\mathcal{A}}$ where

  $$\Sigma = \{!(p,q)\,,\,?(p,q)\,,\,\mathrm{spawn}(p,q) \mid p,q \in \mathbb{N} \text{ with } p \neq q\}$$

- $L_{\mathrm{word}}(\mathcal{A}) \subseteq \Sigma^*$

Dynamic Communicating Automaton

Configuration

Quadruple $(\mathcal{P}, \mathit{state}, \mathit{proc}, \mathit{ch})$ where

- $\mathcal{P} \subseteq \mathbb{N}$ nonempty finite
- $\mathit{state} : \mathcal{P} \to Q$
- $\mathit{proc} : \mathcal{P} \to \mathcal{P}^X$
- $\mathit{ch} : (\mathcal{P} \times \mathcal{P}) \to \left(\mathit{Msg} \times \mathcal{P}^X\right)^*$

Transition system

- initial configuration: $(\{p\}, p \mapsto \iota, (p,x) \mapsto p, (p,p) \mapsto \epsilon)$
- final configuration: all states final and all channels empty
- $\Longrightarrow_{\mathcal{A}} \subseteq \mathit{Conf}_{\mathcal{A}} \times (\Sigma \cup \{\epsilon\}) \times \mathit{Conf}_{\mathcal{A}}$ where

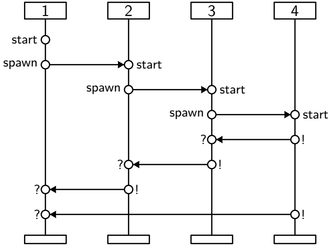
$$\Sigma = \{!(p,q)\,,\ ?(p,q)\,,\ \text{spawn}(p,q) \mid p, q \in \mathbb{N} \text{ with } p \neq q\}$$

- $L_{\text{word}}(\mathcal{A}) \subseteq \Sigma^*$

Message Sequence Charts

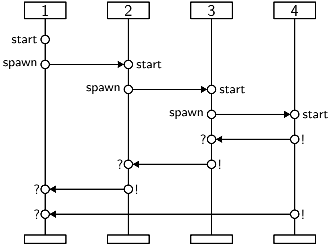
$\mathsf{spawn}(1,2)\ \mathsf{spawn}(2,3)\ \mathsf{spawn}(3,4)\ !(4,3)\ !(4,1)\ ?(4,3)\ !(3,2)\ ?(3,2)\ !(2,1)\ ?(2,1)\ ?(1,4)$

Message Sequence Charts



spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3) !(3, 2) ?(3, 2) !(2, 1) ?(2, 1) ?(1, 4)

Message Sequence Charts



spawn(1, 2) spawn(2, 3) spawn(3, 4) !(4, 3) !(4, 1) ?(4, 3) !(3, 2) ?(3, 2) !(2, 1) ?(2, 1) ?(1, 4)

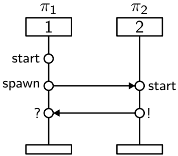
$$L_{\mathsf{word}}(\mathcal{A}) \subseteq \Sigma^*$$

$$L(\mathcal{A}) \subseteq \text{MSCs}$$

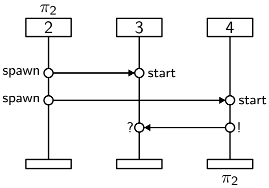
Presentation outline

Building Blocks of Dynamic MSC Grammars

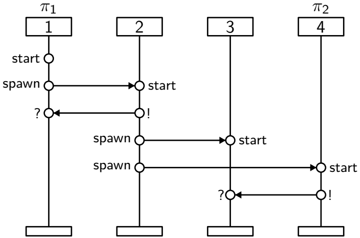
(M_1, ν_1)



(M_2, μ_2)



$(M, \nu) = (M_1, \nu_1) \circ (M_2, \mu_2)$



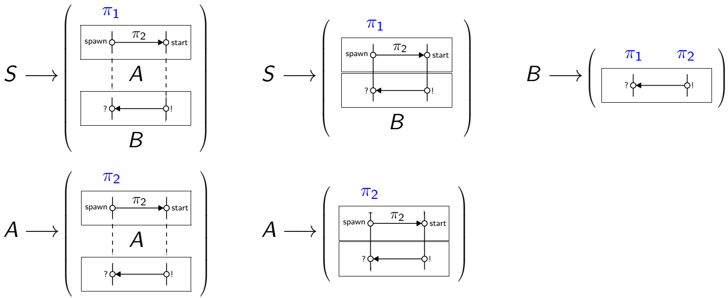
in-out partial MSC

$$\mu_2 : \begin{cases} \Pi & \rightharpoonup \{2,3,4\}^2 \\ \pi_1 & \mapsto \\ \pi_2 & \mapsto (2,4) \end{cases}$$

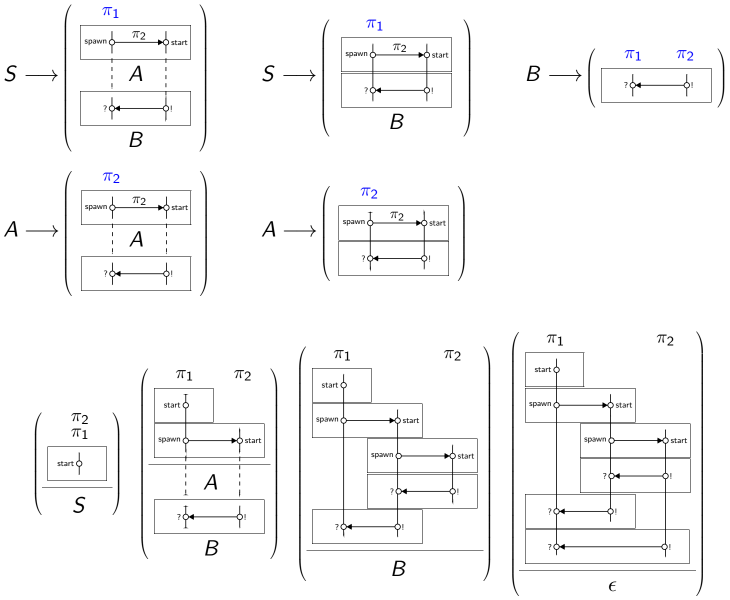
named MSC

$$\nu : \begin{cases} \Pi & \to \mathcal{P} \\ \pi_1 & \mapsto 1 \\ \pi_2 & \mapsto 4 \end{cases}$$

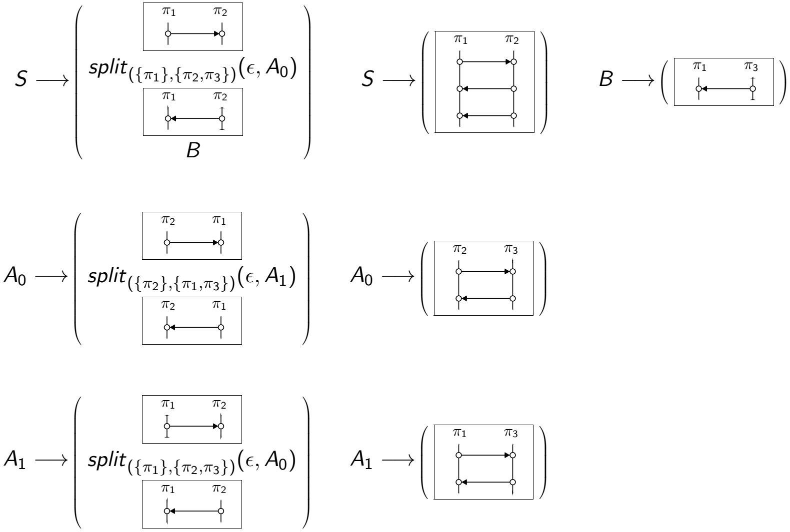
Dynamic MSC Grammar



Dynamic MSC Grammar



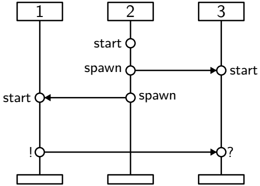
Fork-and-Join Grammar [LMM'02]



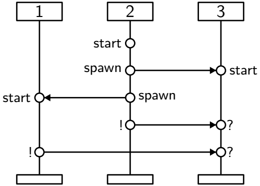
[LMM'02] Leucker & Madhusudan & Mukhopadhyay. Dynamic Message Sequence Charts. 2002.

Presentation outline

Realizability

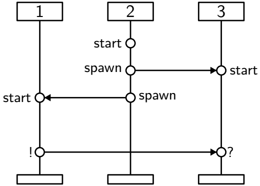


non-realizable

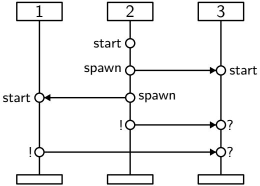


2-realizable

Realizability



non-realizable



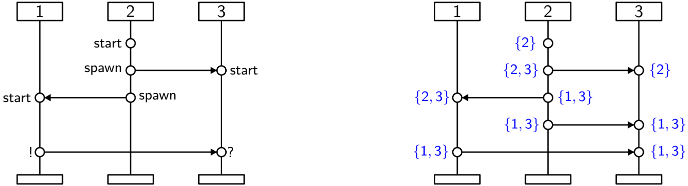
2-realizable

Definition

Let L be a set of MSCs and $b \in \mathbb{N} \cup \{\infty\}$.

- L is realizable if there is a DCA $\mathcal{A}$ such that $L = L(\mathcal{A})$.
- L is *b-realizable* if there is a DCA $\mathcal{A} = (X, Msg, Q, \Delta, \iota, F)$ such that $L = L(\mathcal{A})$ and $|X| \leq b$.

Realizability



non-realizable

2-realizable

Definition

Let L be a set of MSCs and $b \in \mathbb{N} \cup \{\infty\}$.

- L is realizable if there is a DCA $\mathcal{A}$ such that $L = L(\mathcal{A})$.
- L is *b-realizable* if there is a DCA $\mathcal{A} = (X, Msg, Q, \Delta, \iota, F)$ such that $L = L(\mathcal{A})$ and $|X| \leq b$.

Realizability

Theorem

The following problems are decidable:

INPUT: *Dynamic MSC grammar G.*

QUESTION 1: *Is $L(G)$ empty?*

QUESTION 2: *Is $L(G)$ realizable?*

Question 1 can be decided in exponential time.
Question 2 can be decided in doubly exponential time.

Proof

- Build tree automaton $\mathcal{A}_G$ accepting the valid parse trees of G.
- Build tree automaton $\mathcal{B}_G$ accepting the realizable parse trees of G.
- $L(G)$ empty $\iff$ $L(\mathcal{A}_G) = \emptyset$.
- $L(G)$ realizable $\iff$ $L(\mathcal{A}_G) \setminus L(\mathcal{B}_G) = \emptyset$.
- $L(G)$ realizable $\implies$ $L(G)$ $(|Proc(G)| + a \cdot |\Pi|)$-realizable.

Realizability

Theorem

The following problems are decidable:

INPUT: *Dynamic MSC grammar G.*

QUESTION 1: *Is $L(G)$ empty?*

QUESTION 2: *Is $L(G)$ realizable?*

Question 1 can be decided in exponential time.
Question 2 can be decided in doubly exponential time.

Proof

- Build tree automaton $\mathcal{A}_G$ accepting the valid parse trees of G.
- Build tree automaton $\mathcal{B}_G$ accepting the realizable parse trees of G.
- $L(G)$ empty $\iff$ $L(\mathcal{A}_G) = \emptyset$.
- $L(G)$ realizable $\iff$ $L(\mathcal{A}_G) \setminus L(\mathcal{B}_G) = \emptyset$.
- $L(G)$ realizable $\implies$ $L(G)$ $(|Proc(G)| + a \cdot |\Pi|)$-realizable.

Realizability

Theorem

The following problems are decidable:

INPUT: *Dynamic MSC grammar G.*

QUESTION 1: *Is $L(G)$ empty?*

QUESTION 2: *Is $L(G)$ realizable?*

Question 1 can be decided in exponential time.
Question 2 can be decided in doubly exponential time.

Proof

- Build tree automaton $\mathcal{A}_G$ accepting the valid parse trees of G.
- Build tree automaton $\mathcal{B}_G$ accepting the realizable parse trees of G.
- $L(G)$ empty $\iff$ $L(\mathcal{A}_G) = \emptyset$.
- $L(G)$ realizable $\iff$ $L(\mathcal{A}_G) \setminus L(\mathcal{B}_G) = \emptyset$.
- $L(G)$ realizable $\Longrightarrow$ $L(G)$ $(|Proc(G)| + a \cdot |\Pi|)$-realizable.

Realizability

Theorem

The following problems are decidable:

INPUT: *Dynamic MSC grammar G.*

QUESTION 1: *Is $L(G)$ empty?*

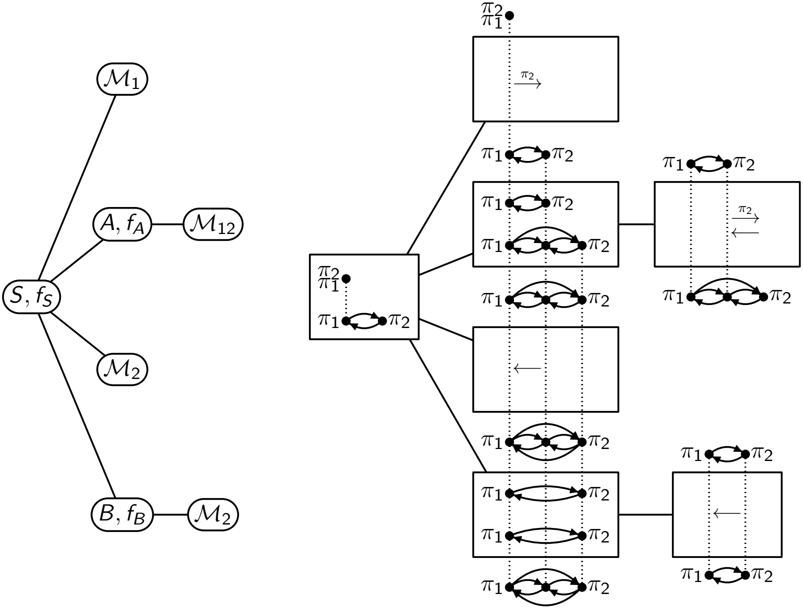
QUESTION 2: *Is $L(G)$ realizable?*

Question 1 can be decided in exponential time.
Question 2 can be decided in doubly exponential time.

Proof

- Build tree automaton $\mathcal{A}_G$ accepting the valid parse trees of G.
- Build tree automaton $\mathcal{B}_G$ accepting the realizable parse trees of G.
- $L(G)$ empty $\iff L(\mathcal{A}_G) = \emptyset$.
- $L(G)$ realizable $\iff L(\mathcal{A}_G) \setminus L(\mathcal{B}_G) = \emptyset$.
- $L(G)$ realizable $\implies L(G)$ $(|Proc(G)| + a \cdot |\Pi|)$-realizable.

The tree automaton



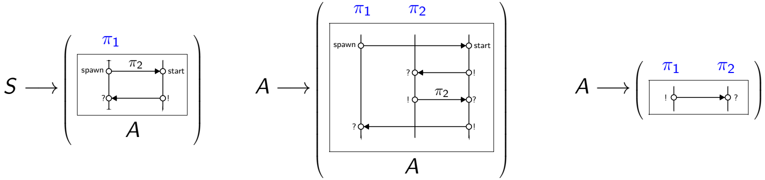
Presentation outline

Local Dynamic MSC Grammars

Definition (Local Grammar, extends [HJ'00 , GMSZ'06])

Every rule is of the form $r = A \longrightarrow_f M.B$ or $A \longrightarrow_f M$ such that

- M has a unique minimal element
- there is $\pi \in Active(r)$ such that, for all B-rules $B \longrightarrow_g \beta$, $M(\beta)$ has a unique minimal element e satisfying $g(loc(e)) = \pi$.



[HJ'00] Hélouët & Jard. Conditions for synthesis of communicating automata from HMSCs. 2000.

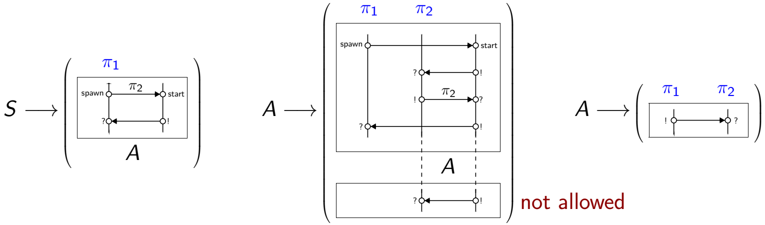
[GMSZ'06] Genest & Muscholl & Seidl & Zeitoun. Infinite-State High-Level MSCs: Model Checking and Realizability. 2006.

Local Dynamic MSC Grammars

Definition (Local Grammar, extends [HJ'00 , GMSZ'06])

Every rule is of the form $r = A \longrightarrow_f M.B$ or $A \longrightarrow_f M$ such that

- M has a unique minimal element
- there is $\pi \in Active(r)$ such that, for all B-rules $B \longrightarrow_g \beta$, $M(\beta)$ has a unique minimal element e satisfying $g(loc(e)) = \pi$.



[HJ'00] Hélouët & Jard. Conditions for synthesis of communicating automata from HMSCs. 2000.
[GMSZ'06] Genest & Muscholl & Seidl & Zeitoun. Infinite-State High-Level MSCs: Model Checking and Realizability. 2006.

Local-choice Dynamic MSC Grammars

Theorem

Let $G = (\Pi, \mathcal{N}, S, \longrightarrow)$ be a local grammar such that $L(G)$ is realizable.

There is a finite DCA $\mathcal{A} = (X, Msg, Q, \Delta, \iota, F)$ such that $L(\mathcal{A}) = L(G)$. Hereby, $|X|$ and $|Msg|$ are polynomial, $|Q|$ and $|Act_{\mathcal{A}}|$ exponential in $|G|$.

Proof.

- x_α for each $\alpha \in \Pi \cup Proc(G)$ ($L(G)$ is $|\Pi| + |Proc(G)|$-realizable)
- local states: $(rule, process, next\_event, next\_rule, Ids)$ and $Poll(Ids)$
- messages: $(rule, (send, receive), next\_rule)$

When process p applies rule r, renaming σ is guessed.
We can assume that the "free processes" of r are known to p. □

Local-choice Dynamic MSC Grammars

Theorem

Let $G = (\Pi, \mathcal{N}, S, \longrightarrow)$ *be a local grammar such that* $L(G)$ *is realizable.*

There is a finite DCA $\mathcal{A} = (X, Msg, Q, \Delta, \iota, F)$ *such that* $L(\mathcal{A}) = L(G)$*.*
Hereby, $|X|$ *and* $|Msg|$ *are polynomial,* $|Q|$ *and* $|Act_{\mathcal{A}}|$ *exponential in* $|G|$*.*

Proof.

- x_α for each $\alpha \in \Pi \cup Proc(G)$ ($L(G)$ is $|\Pi| + |Proc(G)|$-realizable)
- local states: $(rule, process, next\_event, next\_rule, Ids)$ and $Poll(Ids)$
- messages: $(rule, (send, receive), next\_rule)$

When process p applies rule r, renaming σ is guessed.
We can assume that the "free processes" of r are known to p. □

Future work

- More classes of implementable dynamic MSC grammars
- Extended grammars (regular expressions on right-hand sides)
- Dynamic regular MSC languages [HMNST'05]
- DCA and Logic [BL'05 , GKM'06 , HMNST'05]
- Connection with π-calculus, series-parallel languages [LW'00], ...

[BL'06] B. & Leucker. Message-Passing Automata are expressively equivalent to EMSO Logic. 2006.

[GKM'06] Genest & Kuske & Muscholl. A Kleene theorem and model checking algorithms for existentially bounded communicating automata. 2006.

[HMNST'05] Henriksen & Mukund & Narayan Kumar & Sohoni & Thiagarajan. A Theory of Regular MSC Languages. 2005.

[LW'00] Lodaya & Weil. Series-parallel languages and the bounded-width property. 2000.