Theoretical Foundations of the UML

Lecture 7: Communicating Finite-State Machines

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

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

May 11, 2020

Outline

Overview

1. Introduction
2. Communicating Finite-State Machines
3. Semantics of Communicating Finite-State Machines
4. Emptiness Problem for CFMs

Specification to implementation

- consider (C)MSGs as <u>complete</u> specifications of a system
- MSG g, $L(g)$ = set of MSCs = set of possible scenarios

finite

countably infinite (e.g. CMSG for the Yannakakis example)

<u>Central question</u>: can we obtain a system <u>"realisation"</u> that exhibits all possible scenarios in $L(g)$

First question: how do such system "realisations" look like?

- model the behavior of each process by a finite automaton ("local" automaton)
- processes can communicate via unbounded FIFO channels

Specification to implementation

- Consider an MSGs as complete system specifications
  - they describe a full set of possible system scenarios

$L(G)$ = set of all possible scenarios

Specification to implementation

- Consider an MSGs as complete system specifications
  - they describe a full set of possible system scenarios

- Can we obtain “realisations“ that exhibit precisely these scenarios?

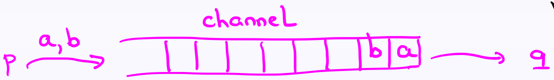
central question in the next 3-4 lectures

Specification to implementation

- Consider an MSGs as complete system specifications
  - they describe a full set of possible system scenarios
- Can we obtain “realisations“ that exhibit precisely these scenarios?
- Map MSGs, i.e., scenarios onto an "executable model"
  - model each process by a finite-state automaton
  - that communicate via unbounded directed FIFO channels

$p \longrightarrow q$

(c) MSG $\longmapsto$ communicating finite-state machine (CFM)



Specification to implementation

- Consider an MSGs as complete system specifications
  - they describe a full set of possible system scenarios
- Can we obtain “realisations“ that exhibit precisely these scenarios?
- Map MSGs, i.e., scenarios onto an executable model
  - model each process by a finite-state automaton
  - that communicate via unbounded directed FIFO channels

⇒ This yields Communicating Finite-state Machines

Brand & Zafiropoulou

Intuition

Example 1

process p "realisation" process q

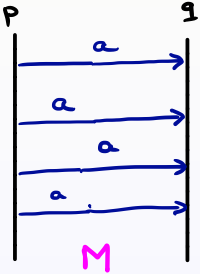
"local" automaton of p

"local" automaton of q

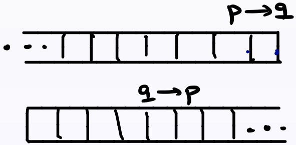
global initial state = (1, A)

global final states = { (1, A) }

CMSC



possible behavior of the CFM:



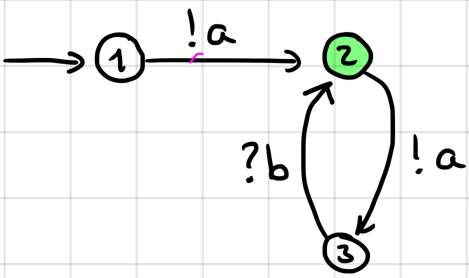
not used

CFM accepts if

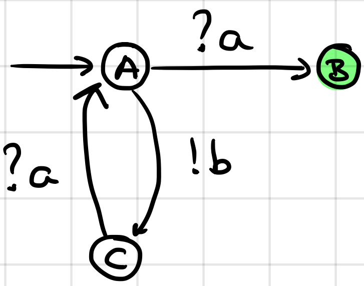
1. all channels are empty
2. we are in state (1, A)

Example 2

process p

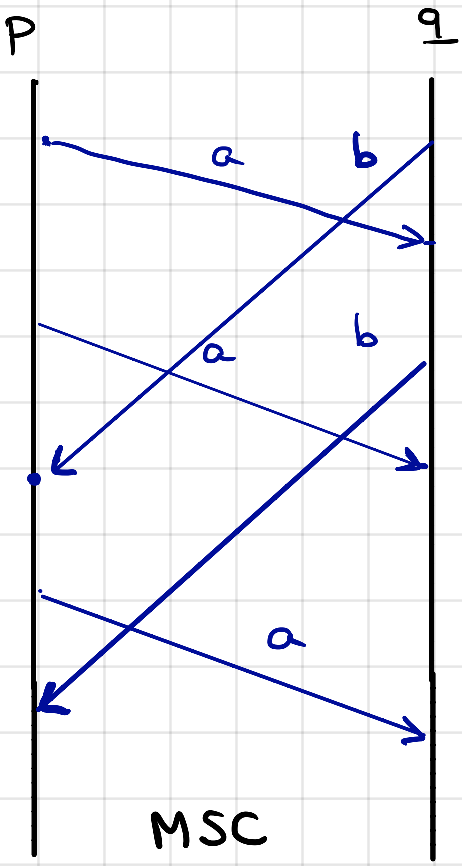


process q

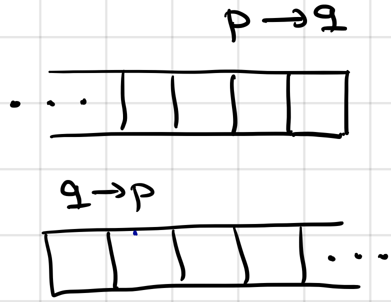


global initial state = $(1, A)$

global final states = $\{(2, B)\}$



MSC

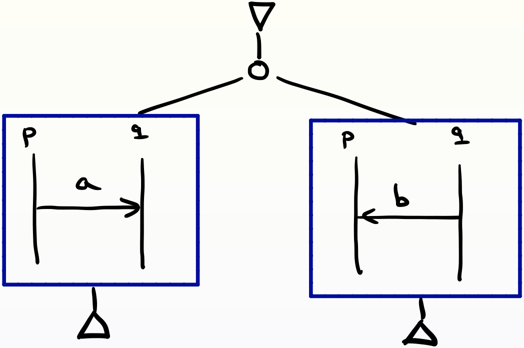


is "accepted" by the example CFM

(Yannakakis example)

The need for synchronisation messages

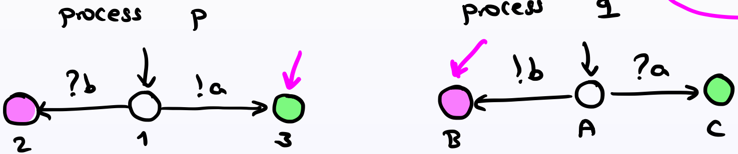
Suppose we want to realise

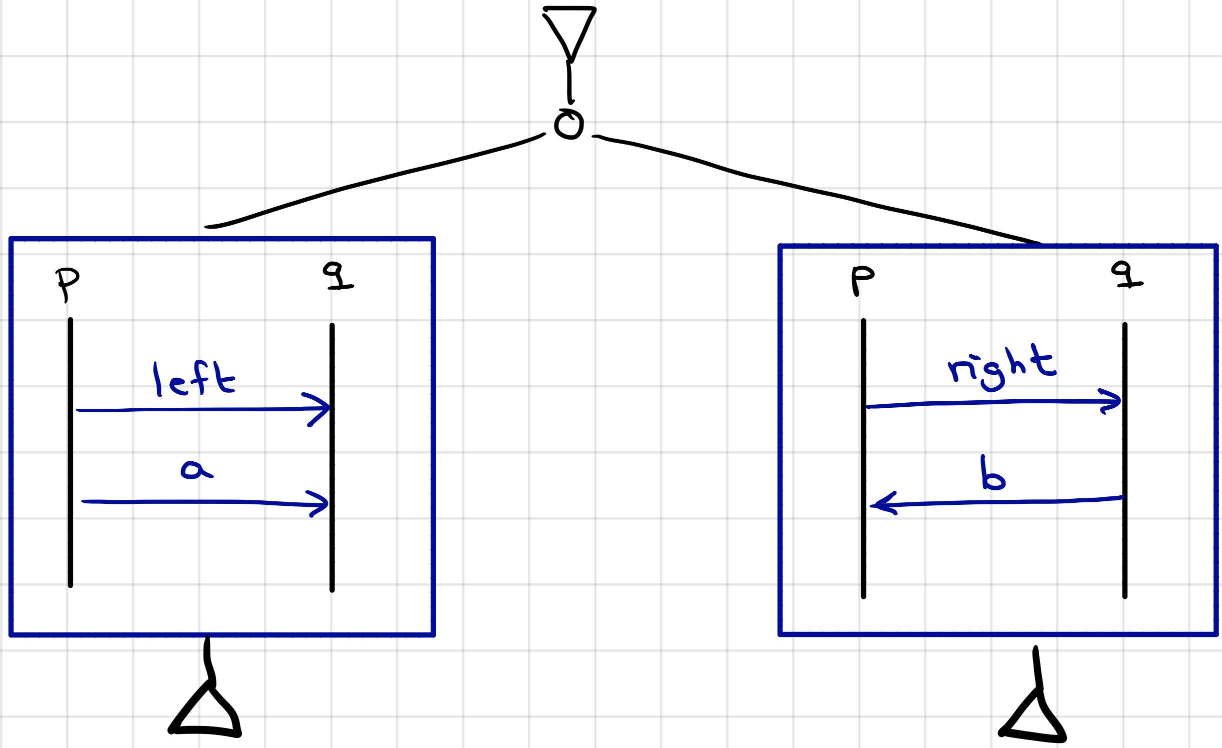


final states
$\{(2,B),(3,C)\}$

CFM has a deadlock

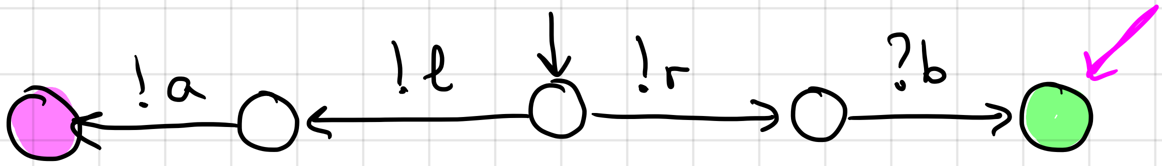
CFM:



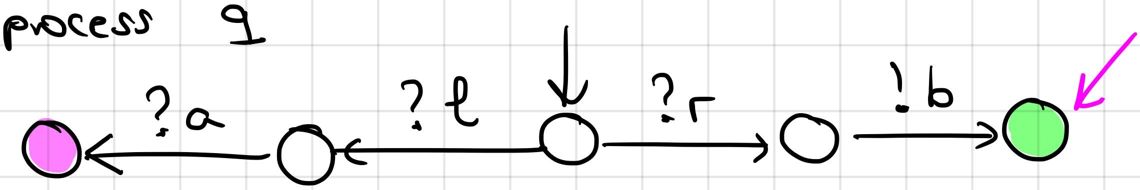


Process p informs process q whether to go "left" or "right"

Automaton for process p: $F = \{(\circ,\circ),(\circ,\circ)\}$



For process q



A deadlock like in the previous example cannot occur

Overview

Preliminaries

Definition

Let

- $\mathcal{P}$ be a finite set of at least two (sequential) processes
- $\mathcal{C}$ be a finite set of message contents

a, b, c

Preliminaries

Definition

Let

- $\mathcal{P}$ be a finite set of at least two (sequential) processes
- $\mathcal{C}$ be a finite set of message contents

Definition (communication actions, channels)

- $Act_p^! := \{!(p, q, a) \mid q \in \mathcal{P} \setminus \{p\}, \ a \in \mathcal{C}\}$
  the set of send actions by process p

Preliminaries

Definition

Let

- $\mathcal{P}$ be a finite set of at least two (sequential) processes
- $\mathcal{C}$ be a finite set of message contents

Definition (communication actions, channels)

- $Act_p^! := \{!(p, q, a) \mid q \in \mathcal{P} \setminus \{p\},\ a \in \mathcal{C}\}$
  the set of send actions by process p
- $Act_p^? := \{?(p, q, a) \mid q \in \mathcal{P} \setminus \{p\},\ a \in \mathcal{C}\}$
  the set of receive actions by process p

Preliminaries

Definition

Let

- $\mathcal{P}$ be a finite set of at least two (sequential) processes
- $\mathcal{C}$ be a finite set of message contents

Definition (communication actions, channels)

- $Act_p^! := \{!(p,q,a) \mid q \in \mathcal{P} \setminus \{p\},\ a \in \mathcal{C}\}$
  the set of send actions by process p
- $Act_p^? := \{?(p,q,a) \mid q \in \mathcal{P} \setminus \{p\},\ a \in \mathcal{C}\}$
  the set of receive actions by process p
- $Act_p := Act_p^! \cup Act_p^?$

Preliminaries

Definition

Let

- $\mathcal{P}$ be a finite set of at least two (sequential) processes
- $\mathcal{C}$ be a finite set of message contents

Definition (communication actions, channels)

- $Act_p^! := \{!(p,q,a) \mid q \in \mathcal{P} \setminus \{p\},\ a \in \mathcal{C}\}$
  the set of send actions by process p
- $Act_p^? := \{?(p,q,a) \mid q \in \mathcal{P} \setminus \{p\},\ a \in \mathcal{C}\}$
  the set of receive actions by process p
- $Act_p := Act_p^! \cup Act_p^?$
- $Act := \bigcup_{p \in \mathcal{P}} Act_p$

Preliminaries

Definition

Let

- $\mathcal{P}$ be a finite set of at least two (sequential) processes
- $\mathcal{C}$ be a finite set of message contents

Definition (communication actions, channels)

- $Act_p^! := \{!(p,q,a) \mid q \in \mathcal{P} \setminus \{p\},\ a \in \mathcal{C}\}$
  the set of send actions by process p
- $Act_p^? := \{?(p,q,a) \mid q \in \mathcal{P} \setminus \{p\},\ a \in \mathcal{C}\}$
  the set of receive actions by process p
- $Act_p := Act_p^! \cup Act_p^?$
- $Act := \bigcup_{p \in \mathcal{P}} Act_p$
- $Ch := \{(p,q) \mid p,q \in \mathcal{P},\ p \neq q\}$ "channels"

ordered

(p,q)

(q,p)

Definition

A communicating finite-state machine (CFM) over $\mathcal{P}$ and $\mathcal{C}$ is a structure

set of global final states

$$\mathcal{A} = (\underbrace{((S_p, \Delta_p))_{p\in\mathcal{P}}}_{\text{"local" automaton for each process}}, \mathbb{D}, s_{init}, F)$$

where

global initial state

synchronisation messages (e.g. left, right)

Communicating finite-state machines

Definition

A communicating finite-state machine (CFM) over $\mathcal{P}$ and $\mathcal{C}$ is a structure

$$\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \underline{\mathbb{D}}, s_{init}, F)$$

where

e.g.
left, right

- $\mathbb{D}$ is a nonempty finite set of synchronization messages (or data)

We often write $s \xrightarrow{\sigma, m}_p s'$ instead of $(s, \sigma, m, s') \in \Delta_p$

Communicating finite-state machines

Definition

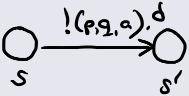
A communicating finite-state machine (CFM) over $\mathcal{P}$ and $\mathcal{C}$ is a structure

$$\mathcal{A} = ((\underline{(S_p, \Delta_p)})_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$$

where

- $\mathbb{D}$ is a nonempty finite set of synchronization messages (or data)
- for each $p \in \mathcal{P}$:
  - S_p is a non-empty finite set of local states (the S_p are disjoint)
  - $\Delta_p \subseteq \underbrace{S_p \times Act_p \times \mathbb{D} \times S_p}$ is a set of local transitions

$$(s, !(p,q,a), d, s') \in \Delta_p$$



$\in Act_p$

We often write $s \xrightarrow{\sigma,m}_p s'$ instead of $\underbrace{(s, \sigma, m, s')} \in \Delta_p$

Communicating finite-state machines

Definition

A communicating finite-state machine (CFM) over $\mathcal{P}$ and $\mathcal{C}$ is a structure

$$\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$$

where

- $\mathbb{D}$ is a nonempty finite set of synchronization messages (or data)
- for each $p \in \mathcal{P}$:
  - S_p is a non-empty finite set of local states (the S_p are disjoint)
  - $\Delta_p \subseteq S_p \times Act_p \times \mathbb{D} \times S_p$ is a set of local transitions
- $s_{init} \in S_{\mathcal{A}}$ is the global initial state
  - where $S_{\mathcal{A}} := \prod_{p \in \mathcal{P}} S_p$ is the set of global states of $\mathcal{A}$

p, q, r

(p, q, r)

We often write $s \xrightarrow{\sigma,m}_p s'$ instead of $(s, \sigma, m, s') \in \Delta_p$

Communicating finite-state machines

Definition

A communicating finite-state machine (CFM) over $\mathcal{P}$ and $\mathcal{C}$ is a structure

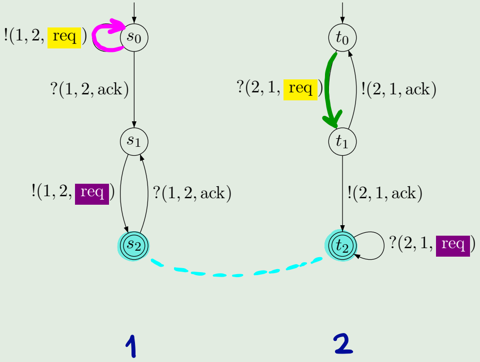
$$\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$$

where

- $\mathbb{D}$ is a nonempty finite set of synchronization messages (or data)
- for each $p \in \mathcal{P}$:
  - S_p is a non-empty finite set of local states (the S_p are disjoint)
  - $\Delta_p \subseteq S_p \times Act_p \times \mathbb{D} \times S_p$ is a set of local transitions
- $s_{init} \in S_{\mathcal{A}}$ is the global initial state
  - where $S_{\mathcal{A}} := \prod_{p \in \mathcal{P}} S_p$ is the set of global states of $\mathcal{A}$
- $F \subseteq S_{\mathcal{A}}$ is the set of global final states

We often write $s \xrightarrow{\sigma,m}_p s'$ instead of $(s, \sigma, m, s') \in \Delta_p$

Example



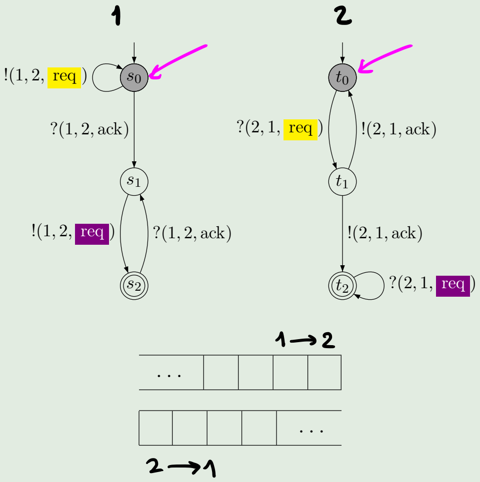
CFM $\mathcal{A}$ over $\mathcal{P} = \{1, 2\}$ and $\mathcal{C} = \{\text{req}, \text{ack}\}$

- $\mathbb{D} = \{\ , \ , \ \}$
- $S_1 = \{s_0, s_1, s_2\}$
- $S_2 = \{t_0, t_1, t_2\}$
- Δ_1: $s_0 \xrightarrow{!(1,2,\text{req})}_1 s_0 \ \ldots$

  Δ_2: $t_0 \xrightarrow{?(2,1,\text{req})}_2 t_1 \ \ldots$
- $s_{init} = (s_0, t_0)$
- $F = \{(s_2, t_2)\}$

Communicating finite-state machines

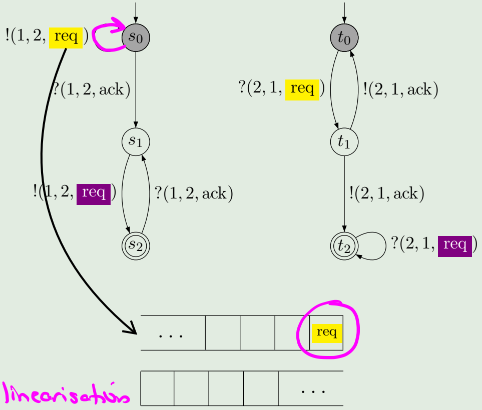
Example

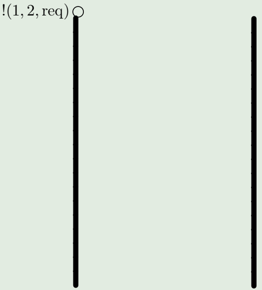


Communicating finite-state machines

Example

CMSC

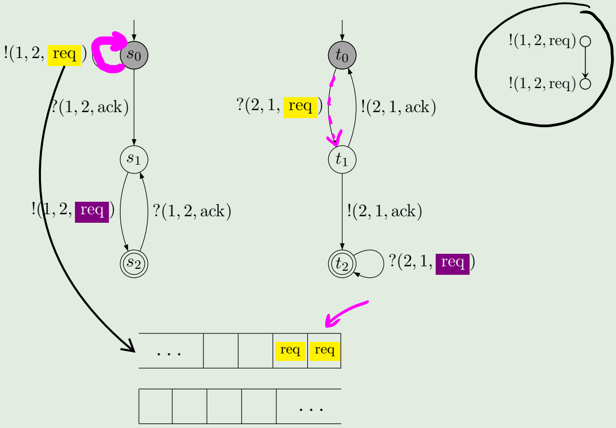




$!(1,2,\text{req})$

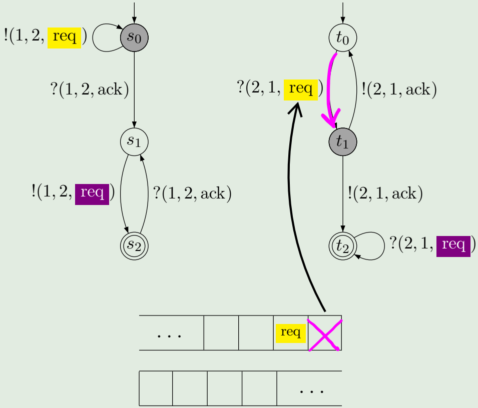
Communicating finite-state machines

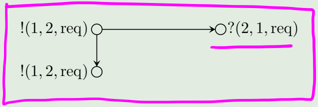
Example



$!(1,2,\text{req})$ $!(1,2,\text{req})$

Example

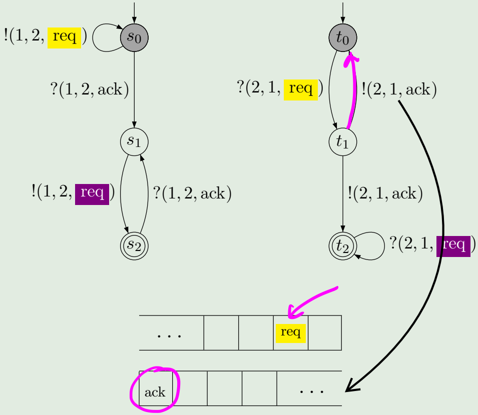




$!(1, 2, \mathrm{req})\ !(1, 2, \mathrm{req})\ ?(2, 1, \mathrm{req})$

Communicating finite-state machines

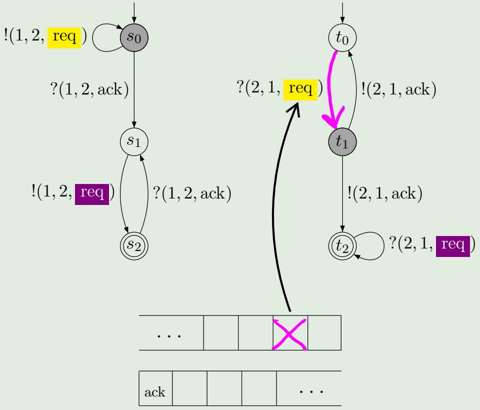
Example



!(1, 2, req) !(1, 2, req) ?(2, 1, req) !(2, 1, ack)

Communicating finite-state machines

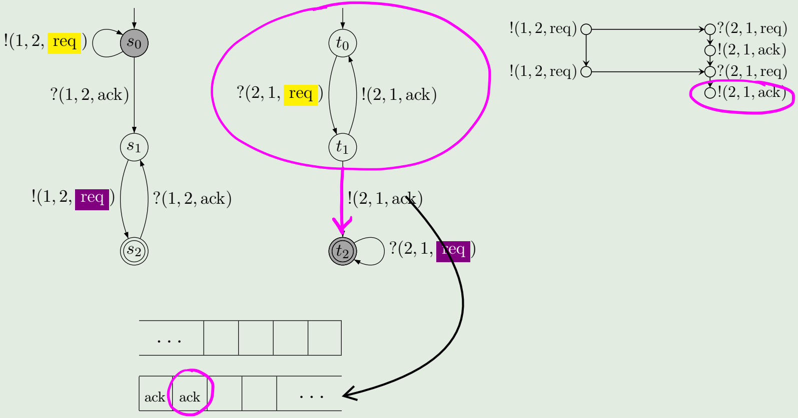
Example



$!(1, 2, \mathrm{req})\ !(1, 2, \mathrm{req})\ ?(2, 1, \mathrm{req})\ !(2, 1, \mathrm{ack})\ ?(2, 1, \mathrm{req})$

Communicating finite-state machines

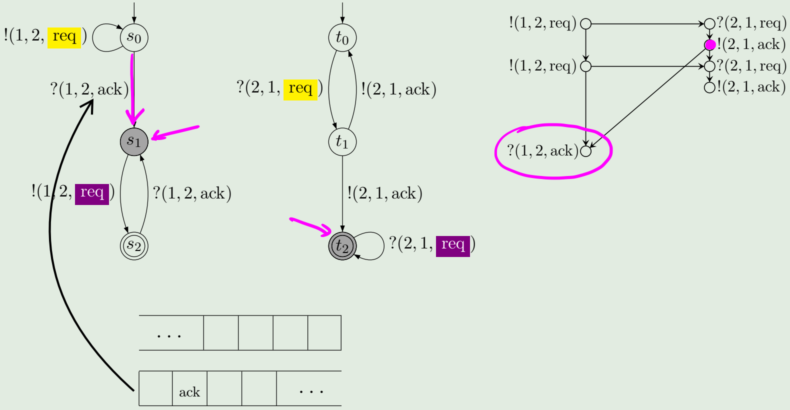
Example



$!(1,2,\text{req})$ $!(1,2,\text{req})$ $?(2,1,\text{req})$ $!(2,1,\text{ack})$ $?(2,1,\text{req})$ $!(2,1,\text{ack})$

Communicating finite-state machines

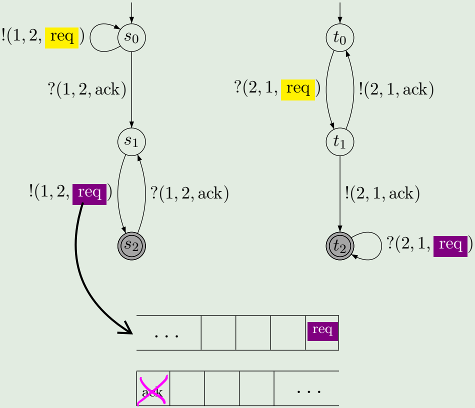
Example

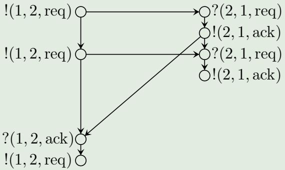


$!(1,2,\text{req})\ !(1,2,\text{req})\ ?(2,1,\text{req})\ !(2,1,\text{ack})\ ?(2,1,\text{req})\ !(2,1,\text{ack})\ ?(1,2,\text{ack})$

Communicating finite-state machines

Example

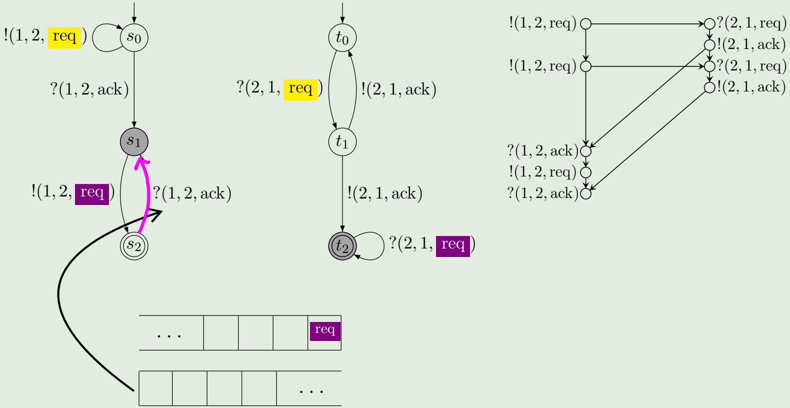




!(1, 2, req) !(1, 2, req) ?(2, 1, req) !(2, 1, ack) ?(2, 1, req) !(2, 1, ack) ?(1, 2, ack) !(1, 2, req)

Communicating finite-state machines

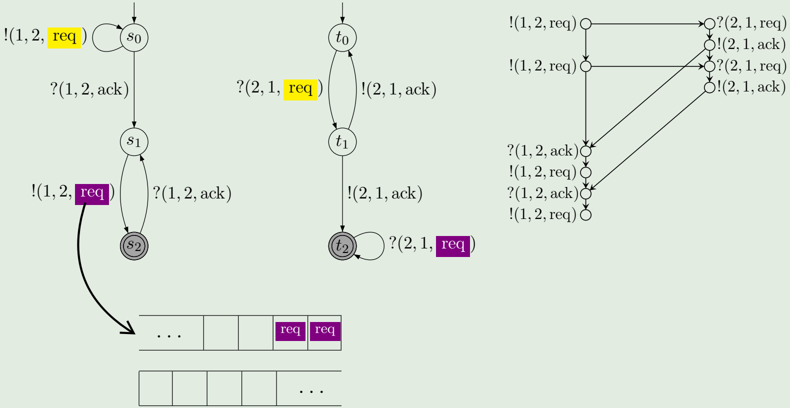
Example



$!(1,2,\mathrm{req})\ !(1,2,\mathrm{req})\ ?(2,1,\mathrm{req})\ !(2,1,\mathrm{ack})\ ?(2,1,\mathrm{req})\ !(2,1,\mathrm{ack})\ ?(1,2,\mathrm{ack})\ !(1,2,\mathrm{req})\ ?(1,2,\mathrm{ack})$

Communicating finite-state machines

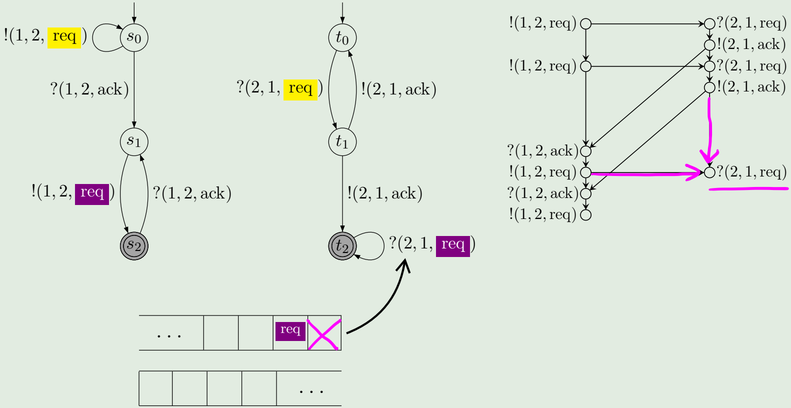
Example



!(1, 2, req) !(1, 2, req) ?(2, 1, req) !(2, 1, ack) ?(2, 1, req) !(2, 1, ack) ?(1, 2, ack) !(1, 2, req) ?(1, 2, ack) !(1, 2, req)

Communicating finite-state machines

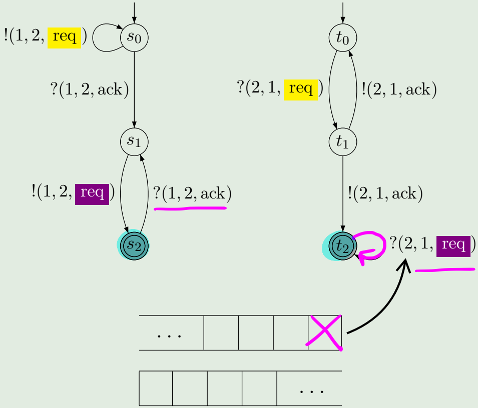
Example

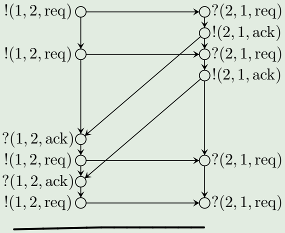


$!(1,2,\text{req})$ $!(1,2,\text{req})$ $?(2,1,\text{req})$ $!(2,1,\text{ack})$ $?(2,1,\text{req})$ $!(2,1,\text{ack})$ $?(1,2,\text{ack})$ $!(1,2,\text{req})$ $?(1,2,\text{ack})$ $!(1,2,\text{req})$ $?(2$

Communicating finite-state machines

Example

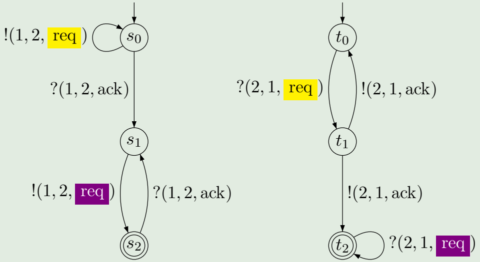


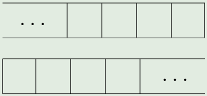


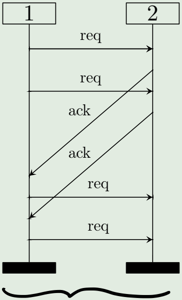
$!(1,2,\text{req})\ !(1,2,\text{req})\ ?(2,1,\text{req})\ !(2,1,\text{ack})\ ?(2,1,\text{req})\ !(2,1,\text{ack})\ ?(1,2,\text{ack})\ !(1,2,\text{req})\ ?(1,2,\text{ack})\ !(1,2,\text{req})\ ?(2$

Communicating finite-state machines

Example







MSC M

M is accepted by this CFM.

$!(1,2,\text{req})\ !(1,2,\text{req})\ ?(2,1,\text{req})\ !(2,1,\text{ack})\ ?(2,1,\text{req})\ !(2,1,\text{ack})\ ?(1,2,\text{ack})\ !(1,2,\text{req})\ ?(1,2,\text{ack})\ !(1,2,\text{req})\ ?(2$

Overview

Formal semantics of CFMs

Let $\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$ be a CFM over $\mathcal{P}$ and $\mathcal{C}$.

Definition (configurations)

Configurations of $\mathcal{A}$: $Conf_{\mathcal{A}} := S_{\mathcal{A}} \times \underbrace{\{\eta \mid \eta : Ch \rightarrow (\mathcal{C} \times \mathbb{D})^*\}}$

global state (= a local state for every process p)

"the content of all channels of the CFM"

$\eta : Ch \rightarrow (C \times D)^*$

$\eta((p,q)) = \varepsilon$

$\eta((p,q)) = (a, \,), (b, \,)$

Formal semantics of CFMs

Let $\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$ be a CFM over $\mathcal{P}$ and $\mathcal{C}$.

Definition (configurations)

Configurations of $\mathcal{A}$: $Conf_{\mathcal{A}} := S_{\mathcal{A}} \times \{\eta \mid \eta : Ch \to (\mathcal{C} \times \mathbb{D})^*\}$

Definition (global step)

$\Longrightarrow_{\mathcal{A}} \subseteq Conf_{\mathcal{A}} \times Act \times \mathbb{D} \times Conf_{\mathcal{A}}$ is defined as follows:

$|P| = k$

$((s_1, \ldots, s_i, \ldots, s_k), \eta) \xrightarrow{!(s_i, s_j, a),\, m} ((s_1, \ldots, s_i', \ldots, s_k), \eta')$

$Ch \to (C \times D)^*$

$?(s_i, s_j, a), m$

Formal semantics of CFMs

Let $\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$ be a CFM over $\mathcal{P}$ and $\mathcal{C}$.

Definition (configurations)

Configurations of $\mathcal{A}$: $Conf_{\mathcal{A}} := S_{\mathcal{A}} \times \{\eta \mid \eta : Ch \to (\mathcal{C} \times \mathbb{D})^*\}$

Definition (global step)

$\Longrightarrow_{\mathcal{A}} \subseteq Conf_{\mathcal{A}} \times Act \times \mathbb{D} \times Conf_{\mathcal{A}}$ is defined as follows: $|P| = k$

- sending a message: $((\overline{s}, \eta), \underbrace{!(p,q,a), m}, (\overline{s}', \eta')) \in \Longrightarrow_{\mathcal{A}}$ if
  - ✓ $(\overline{s}[p], !(p,q,a), m, \overline{s}'[p]) \in \Delta_p$
  - ✓ $\eta' = \eta[(p,q) := (a,m) \cdot \eta((p,q))]$
  - ✓ $\overline{s}[r] = \overline{s}'[r]$ for all $r \in \mathcal{P} \setminus \{p\}$

$\overline{s}[p] = s_p$ local state at process p

$s_p \xrightarrow{!(p,q,a),m} s'_p$

(p,q): $(a,m) \times \times \times \times$ $\underbrace{\quad}_{\eta}$

$(\overline{s} = (s_1, \ldots s_p \ldots, s_k), \eta)$

$\downarrow \; !(p,q,a), m$

$(\overline{s}' = \underbrace{(s_1, \ldots s'_p, \ldots s_k)}, \eta')$

Formal semantics of CFMs

Let $\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$ be a CFM over $\mathcal{P}$ and $\mathcal{C}$.

Definition (configurations)

Configurations of $\mathcal{A}$: $Conf_{\mathcal{A}} := S_{\mathcal{A}} \times \{\eta \mid \eta : Ch \rightarrow (\mathcal{C} \times \mathbb{D})^*\}$

Definition (global step)

$\Longrightarrow_{\mathcal{A}} \subseteq Conf_{\mathcal{A}} \times Act \times \mathbb{D} \times Conf_{\mathcal{A}}$ is defined as follows:

- sending a message: $((\overline{s}, \eta), !(p, q, a), m, (\overline{s}', \eta')) \in \Longrightarrow_{\mathcal{A}}$ if
  - $(\overline{s}[p], !(p, q, a), m, \overline{s}'[p]) \in \Delta_p$
  - $\eta' = \eta[(p, q) := (a, m) \cdot \eta((p, q))]$
  - $\overline{s}[r] = \overline{s}'[r]$ for all $r \in \mathcal{P} \setminus \{p\}$
- receipt of a message: $((\overline{s}, \eta), ?(p, q, a), m, (\overline{s}', \eta')) \in \Longrightarrow_{\mathcal{A}}$ if
  - $(\overline{s}[p], ?(p, q, a), m, \overline{s}'[p]) \in \Delta_p$
  - $\eta((q, p)) = w \cdot (a, m) \neq \epsilon$ and $\eta' = \eta[(q, p) := w]$
  - $\overline{s}[r] = \overline{s}'[r]$ for all $r \in \mathcal{P} \setminus \{p\}$

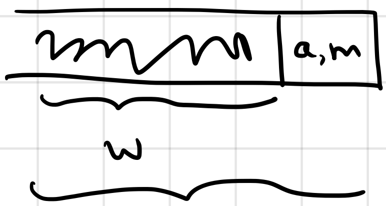
<u>receipt of a message:</u> $((\bar{s},\eta), ?(p,q,a), m, (\bar{s}',\eta')) \in \Longrightarrow_{\mathcal{A}}$ if

- ✓ $(\bar{s}[p], ?(p,q,a), m, \bar{s}'[p]) \in \Delta_p$
- ✓ $\eta((q,p)) = w \cdot (a,m) \neq \epsilon$ and $\eta' = \eta[(q,p) := w]$
- ✓ $\bar{s}[r] = \bar{s}'[r]$ for all $r \in \mathcal{P} \setminus \{p\}$

$$\begin{array}{c} ((\underbrace{s_1,\ldots\ldots}, s_p, \underbrace{\ldots\ldots, s_k}), \eta) \\ \Big\downarrow \; ?(p,q,a),\, m \\ ((\underbrace{s_1,\ldots\ldots}, s_p', \underbrace{\ldots\ldots, s_k}), \eta') \end{array}$$

current configuration

$q \rightarrow p$



$\neq \varepsilon$

$\eta'((q,p)) = w$

for all other channels c

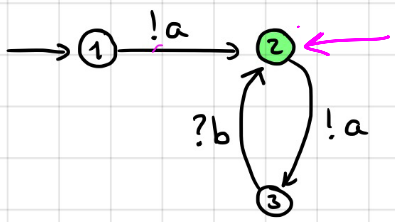
$\eta'(c) = \eta(c)$

$$s_p \xrightarrow{?(p,q,a),\, m} s_p'$$

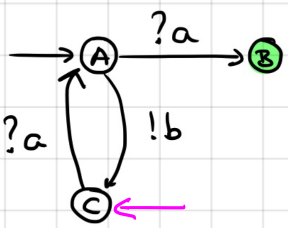
Example

Example 2

process p



process q



global initial state = $(1, A)$

global final states = $\{(2, B)\}$

$\eta((p,q)) = \varepsilon$

$\eta((q,p)) = \varepsilon$

starting configuration: $((1,A),(\varepsilon,\varepsilon)) = \gamma_0$

$$\Downarrow \text{empty}$$

$$((1,C),(\varepsilon,b)) = \gamma_1$$

$$\Downarrow \text{empty}$$

$$((2,C),(a,b)) = \gamma_2$$

$$\Downarrow \text{--}$$

$\eta((p,q)) = aa$

$\eta((q,p)) = b$

$$\ldots \underbrace{((2,C),(aa,\varepsilon))}_{=\gamma_4} \Leftarrow ((3,C),(aa,b)) = \gamma_3$$

Linearizations of a CFM

$!(p,q,a)$, $?(q,p,b)$

Let $\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$ be a CFM over $\mathcal{P}$ and $\mathcal{C}$.

Definition (accepting runs)

A run ρ of CFM $\mathcal{A}$ on word $w = \sigma_1 \ldots \sigma_n \in Act^*$ is an alternating sequence $\rho = \gamma_0\, m_1\, \gamma_1 \ldots \gamma_{n-1}\, m_n\, \gamma_n$ such that

1. $\gamma_0 = (s_{init}, \eta_\varepsilon)$ with η_ε mapping any channel to ε (empty content)
2. $\gamma_{i-1} \xRightarrow{\sigma_i, m_i}_{\mathcal{A}} \gamma_i$ for any $i \in \{1, \ldots, n\}$

Let $\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$ be a CFM over $\mathcal{P}$ and $\mathcal{C}$.

Definition (accepting runs)

A run ρ of CFM $\mathcal{A}$ on word $w = \sigma_1 \ldots \sigma_n \in Act^*$ is an alternating sequence $\rho = \gamma_0\, m_1\, \gamma_1 \ldots \gamma_{n-1}\, m_n\, \gamma_n$ such that

1. $\gamma_0 = (s_{init}, \eta_\varepsilon)$ with η_ε mapping any channel to ε
2. $\gamma_{i-1} \xRightarrow{\sigma_i, m_i}_{\mathcal{A}} \gamma_i$ for any $i \in \{1, \ldots, n\}$

The run ρ is accepting if $\underline{\gamma_n} \in \underline{F} \times \underline{\{\eta_\varepsilon\}}$.

γ_n = global final state + all channels are empty.

Linearizations of a CFM

Let $\mathcal{A} = (((S_p, \Delta_p))_{p \in \mathcal{P}}, \mathbb{D}, s_{init}, F)$ be a CFM over $\mathcal{P}$ and $\mathcal{C}$.

Definition (accepting runs)

A run ρ of CFM $\mathcal{A}$ on word $w = \sigma_1 \ldots \sigma_n \in Act^*$ is an alternating sequence $\rho = \gamma_0 \, m_1 \, \gamma_1 \ldots \gamma_{n-1} \, m_n \, \gamma_n$ such that

1. $\gamma_0 = (s_{init}, \eta_\varepsilon)$ with η_ε mapping any channel to ε
2. $\gamma_{i-1} \xRightarrow{\sigma_i, m_i}_{\mathcal{A}} \gamma_i$ for any $i \in \{1, \ldots, n\}$

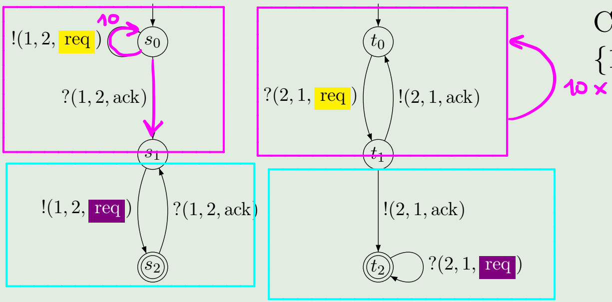
The run ρ is accepting if $\gamma_n \in F \times \{\eta_\varepsilon\}$.

Definition (linearization of a CFM)

The (word) language of CFM $\mathcal{A}$ is defined by:

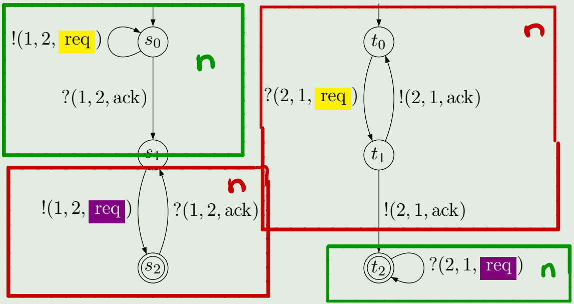
$Lin(\mathcal{A}) := \{\underline{w \in Act^*} \mid \underline{\text{there is an accepting run of } \mathcal{A} \text{ on } w}\}$

Example



CFM $\mathcal{A}$ over $\{1,2\}$ and $\{\text{req},\text{ack}\}$

Example



CFM $\mathcal{A}$ over $\{1, 2\}$ and $\{\text{req}, \text{ack}\}$

w↾1 = the sequence of actions in w that occur at process 1

w↾2 = ... for process 2..

$$Lin(\mathcal{A}) = \Big\{ w \in Act^* \mid \text{there is } n \geqslant 1 \text{ such that:}$$

$$w \restriction 1 = \,!(1,2,\text{req}))^n \; (?(1,2,\text{ack}) \; !(1,2,\text{req}))^n \qquad (*)$$

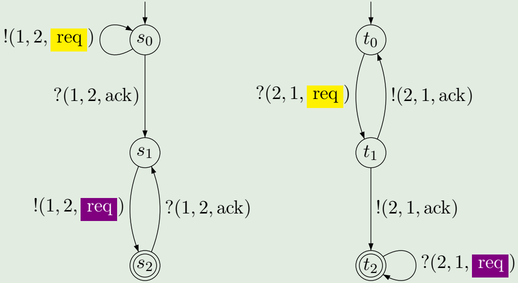
$$w \restriction 2 = (?(2,1,\text{req}) \; !(2,1,\text{ack}))^n \; (?(2,1,\text{req}))^n \qquad (**)$$

for any $u \in Pref(w)$ and $(p, q) \in Ch$:

$$\sum_{a \in \mathcal{C}} |u|_{!(p,q,a)} - \sum_{a \in \mathcal{C}} |u|_{?(q,p,a)} \geqslant 0 \Big\} \qquad (***)$$

Linearizations of an example CFM

Example



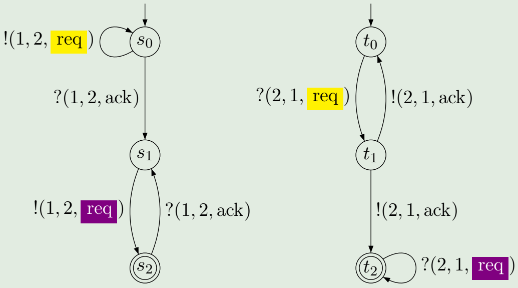
CFM $\mathcal{A}$ over $\{1, 2\}$ and $\{\text{req}, \text{ack}\}$

- $!(1, 2, \text{req})$ and $!(2, 1, \text{ack})$ are always independent.
- $!(1, 2, \text{req})$ and $?(1, 2, \text{ack})$ are always dependent.
- $!(1, 2, \text{req})$ and $?(2, 1, \text{req})$ are sometimes independent.

$\leadsto$ non-regular (word) languages → more expressive than finite-state automata!

Linearizations and MSCs of an example CFM

Example



CFM $\mathcal{A}$ over $\{1,2\}$ and $\{\text{req}, \text{ack}\}$

$Lin(\mathcal{A}) = \Big\{ w \in Act^* \mid$ there is $n \geqslant 1$ such that:

$$w \upharpoonright 1 = (!(1,2,\text{req}))^n \; (?(1,2,\text{ack}) \; !(1,2,\text{req}))^n$$

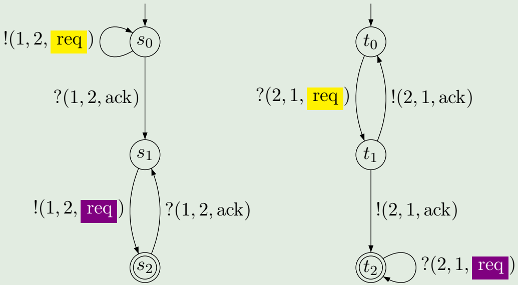
$$w \upharpoonright 2 = (?(2,1,\text{req}) \; !(2,1,\text{ack}))^n \; (?(2,1,\text{req}))^n$$

for any $u \in Pref(w)$ and $(p,q) \in Ch$:

$$\sum_{a \in \mathcal{C}} |u|_{!(p,q,a)} - \sum_{a \in \mathcal{C}} |u|_{?(q,p,a)} \geqslant 0 \Big\}$$

Linearizations and MSCs of an example CFM

Example



CFM $\mathcal{A}$ over $\{1, 2\}$ and $\{\text{req}, \text{ack}\}$

$\mathcal{L}(\mathcal{A}) = \Big\{ M \in \mathbb{M} \mid$ there is $n \geq 1$ such that:

$$M \upharpoonright 1 = \overbrace{(!(1,2,\text{req}))^n}^{s_0} \; \overbrace{(?(1,2,\text{ack}) \; !(1,2,\text{req}))^n}^{s_1 s_2}$$

$$M \upharpoonright 2 = \underbrace{(?(2,1,\text{req}) \; !(2,1,\text{ack}))^n}_{t_0 t_1} \; \underbrace{(?(2,1,\text{req}))^n}_{t_2} \Big\}$$

set of MSCs accepted by CFM A.

Overview

CFMs are more expressive than finite-state automata

does a CFM accept at least one word? undecidable ☹

Elementary questions are undecidable for CFMs

Emptiness of CFMs is undecidable [Brand & Zafiropulo 1983]

The following problem is undecidable (even if $\mathcal{C}$ is a singleton):

INPUT:	CFM $\mathcal{A}$ over processes $\mathcal{P}$ and message contents $\mathcal{C}$
QUESTION:	Is $\mathcal{L}(\mathcal{A})$ empty?

e.g. $C = \{a\}$

the set of MSCs accepted by CFM A

$\updownarrow$

the set of linearisations accepted by CFM A.

Lin(A)

Elementary questions are undecidable for CFMs

Emptiness of CFMs is undecidable [Brand & Zafiropulo 1983]

The following problem is undecidable (even if $\mathcal{C}$ is a singleton):

INPUT: CFM $\mathcal{A}$ over processes $\mathcal{P}$ and message contents $\mathcal{C}$
QUESTION: Is $\mathcal{L}(\mathcal{A})$ empty?

Proof (sketch)

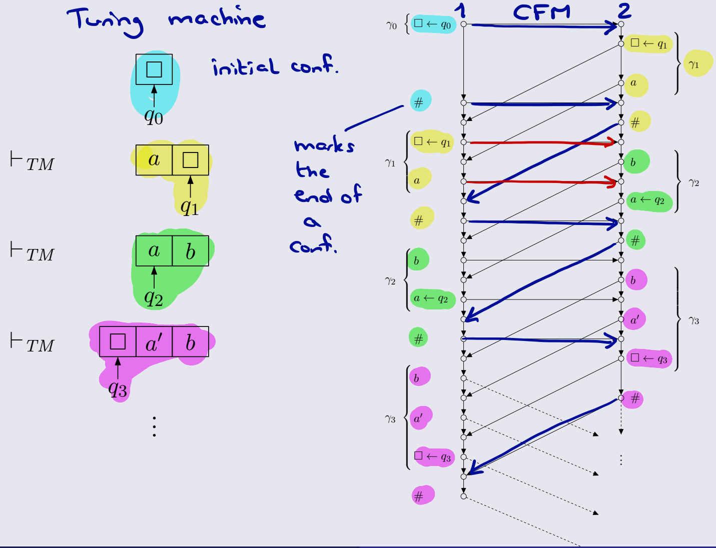
Reduction from the halting problem for Turing machine $TM = (Q, \Sigma, \Delta, \Box, q_0, q_f)$ to emptiness for a CFM with two processes.
Build CFM $\mathcal{A} = ((\mathcal{A}_1, \mathcal{A}_2), \mathbb{D}, s_{init}, F)$ over $\{1, 2\}$ and some singleton set $\mathcal{C}$ such that $\boxed{\mathcal{L}(\mathcal{A}) \neq \emptyset \text{ iff } TM \text{ can reach } q_f\text{, i.e., } TM \text{ accepts.}}$

- Process 1 sends current configurations to process 2 ✓
- Process 2 chooses successor configurations and sends them to 1 ✓
- $\mathbb{D} = \Big((\Sigma \cup \{\Box\}) \times (Q \cup \{\_\})\Big) \cup \{\#\}$

of the TM

A CFM simulating a Turing machine

Proof (contd.)



A CFM simulating a Turing machine

Proof (contd.)

- Left or standstill transition: Process 2 may just wait for a symbol containing a state of TM and to alter it correspondingly. In the example, the left-moving transition (q_2, a, a', L, q_3) is applied so that process 2
  - sends b unchanged back to process 1
  - detects (receives) $a \leftarrow q_2$
  - sends a' to process 1 entering a state indicating that the symbol to be sent next has to be equipped with q_3
  - receives $\#$ so that the symbol $\Box \leftarrow q_3$ has to be inserted before returning $\#$

A CFM simulating a Turing machine

Proof (contd.)

- Left or standstill transition: Process 2 may just wait for a symbol containing a state of TM and to alter it correspondingly. In the example, the left-moving transition (q_2, a, a', L, q_3) is applied so that process 2
  - sends b unchanged back to process 1
  - detects (receives) $a \leftarrow q_2$
  - sends a' to process 1 entering a state indicating that the symbol to be sent next has to be equipped with q_3
  - receives $\#$ so that the symbol $\Box \leftarrow q_3$ has to be inserted before returning $\#$
- Right transition: Process 2 has to guess what the position right before the head is. For example, provided process 2 decided in favor of (q_2, a, a', R, q_3) while reading b, it would have to
  - send $b \leftarrow q_3$ instead of just b, entering some state $t(a \leftarrow q_2)$
  - receive $a \leftarrow q_2$ (no other symbol can be received in state $t(a \leftarrow q_2)$)
  - send a' back to process 1

Communicating Finite-state Machines

"realisation" of system

operational model of an implementation

(c) MSG = "requirements"

all scenarios a system should exhibit

A CFM simulating a Turing machine

Proof (contd.)

- Introduce local final states s_f and t_f, one for process 1 and one for process 2, respectively (i.e., $F = \{(s_f, t_f)\}$ and $\mathcal{A}$ is locally accepting).
- At any time, process 1 may switch into s_f, in which arbitrary and arbitrarily many messages can be received to empty channel $(2, 1)$.
- Process 2 is allowed to move into t_f and to empty the channel $(1, 2)$ as soon as it receives a letter $c \leftarrow q_f$ for some c.
- As process 2 modifies a configuration of TM locally, finitely many states are sufficient in $\mathcal{A}$. $\square$