Semantics of swarms

By rule [Local] below, a command's execution updates both local and global logs

$$\frac{\mathtt{S}(i) = \mathtt{M}\boxed{\ell_i} \qquad \mathtt{M}\boxed{\ell_i} \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} \mathtt{M}\boxed{\ell_i'} \qquad \mathit{src}(\ell_i' \setminus \ell_i) = \{i\} \qquad \ell' \in \ell \bowtie \ell_i'}{(\mathtt{S}, \ell) \xrightarrow{\mathtt{c}\,/\,\mathtt{l}} (\mathtt{S}[i \mapsto \mathtt{M}\boxed{\ell_i'}], \ell')}\,[\text{Local}]$$

$$\frac{\mathtt{S}(i) = \mathtt{M}\boxed{\ell_i} \qquad \ell_i \sqsubseteq \ell' \sqsubseteq \ell \qquad \ell_i \subset \ell'}{(\mathtt{S}, \ell) \xrightarrow{\tau} (\mathtt{S}[i \mapsto \mathtt{M}\boxed{\ell'}], \ell)}\,[\text{Prop}]$$

By rule [Prop] above, the propagation of events happens

- by shipping a **non-deterministically chosen** subset of events in the global log
- to a **non-deterministically chosen** machine

Semantics at work (I)

If $\mathtt{B}\boxed{b} \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{B}\boxed{b \cdot d \cdot e}$ with $\vdash d \cdot e : \mathtt{1}$

Semantics at work (I)

If $\mathtt{B}\boxed{b} \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{B}\boxed{b \cdot d \cdot e}$ with $\vdash d \cdot e : \mathtt{1}$

then, by [Local] $\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{c} \mid b \cdot a \cdot c \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell$

Semantics at work (I)

If $\mathtt{B}\boxed{b} \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{B}\boxed{b \cdot d \cdot e}$ with $\vdash d \cdot e : \mathtt{1}$

then, by [Local] $\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{c} \mid b \cdot a \cdot c \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell$

for all $\ell \in (b \cdot a \cdot c) \bowtie (b \cdot d \cdot e)$

Semantics at work (I)

If $\mathtt{B}\boxed{b} \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{B}\boxed{b \cdot d \cdot e}$ with $\vdash d \cdot e : \mathtt{1}$

then, by [Local] $\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{c} \mid b \cdot a \cdot c \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell$

for all $\ell \in (b \cdot a \cdot c) \bowtie (b \cdot d \cdot e)$

Exercise

Compute $(b \cdot a \cdot c) \bowtie (b \cdot d \cdot e)$

Semantics at work (I)

If $\mathtt{B}\boxed{b} \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{B}\boxed{b \cdot d \cdot e}$ with $\vdash d \cdot e : \mathtt{1}$

then, by [Local] $\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{c} \mid b \cdot a \cdot c \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell$

for all $\ell \in (b \cdot a \cdot c) \bowtie (b \cdot d \cdot e)$

Exercise

Compute $(b \cdot a \cdot c) \bowtie (b \cdot d \cdot e)$

$$\{\ b \cdot a \cdot c \cdot d \cdot e,\ b \cdot a \cdot d \cdot c \cdot e,\ b \cdot a \cdot d \cdot e \cdot c,\ b \cdot d \cdot a \cdot e \cdot c,\ b \cdot d \cdot a \cdot c \cdot e,\ b \cdot d \cdot e \cdot a \cdot c\ \}$$

Semantics at work (II)

Exercise

Is $\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{} \mid b \xrightarrow{\tau} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{} \mid a \cdot b$ possible? Does it actually make sense?

Semantics at work (II)

Exercise

Is $\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{} \mid b \xrightarrow{\tau} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{} \mid a \cdot b$ possible? Does it actually make sense?

With reference to slide 56, consider

$$\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{c} \mid b \cdot a \cdot c \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \overbrace{b \cdot a \cdot d \cdot e \cdot c}^{=\ell}$$

Semantics at work (II)

Exercise

Is $\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{} \mid b \xrightarrow{\tau} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{} \mid a \cdot b$ possible? Does it actually make sense?

With reference to slide 56, consider

$$\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{c} \mid b \cdot a \cdot c \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \overbrace{b \cdot a \cdot d \cdot e \cdot c}^{=\ell}$$

Let's propagate $d \cdot e$: by [Prop] we have to non-deterministically choose a sublog of ℓ

$$\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell \begin{cases} \xrightarrow{\tau} \mathtt{A}\boxed{b \cdot a \cdot d \cdot e} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell \\ \xrightarrow{\tau} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{b \cdot d \cdot e \cdot c} \mid \ell \end{cases}$$

b is shipped too 'cause sublogs must be downward wrt emitting machines

Semantics at work (II)

Exercise

Is $\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{} \mid b \xrightarrow{\tau} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{} \mid a \cdot b$ possible? Does it actually make sense?

With reference to slide 56, consider

$$\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b} \mid \mathtt{C}\boxed{c} \mid b \cdot a \cdot c \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \overbrace{b \cdot a \cdot d \cdot e \cdot c}^{=\ell}$$

Let's propagate $d \cdot e$: by [Prop] we have to non-deterministically choose a sublog of ℓ

$$\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell \xrightarrow{\tau} \mathtt{A}\boxed{b \cdot a \cdot d \cdot e} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell$$

$$\mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{c} \mid \ell \xrightarrow{\tau} \mathtt{A}\boxed{a} \mid \mathtt{B}\boxed{b \cdot d \cdot e} \mid \mathtt{C}\boxed{b \cdot d \cdot e \cdot c} \mid \ell$$

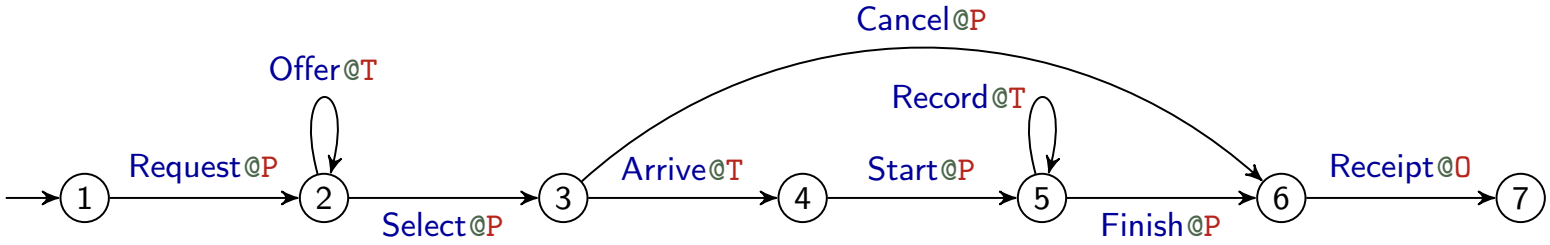
b is shipped too 'cause sublogs must be downward wrt emitting machines

Exercise

Can we propagate just event e?
Can event a be shipped to $\mathtt{C}$ together with events from $\mathtt{B}$?

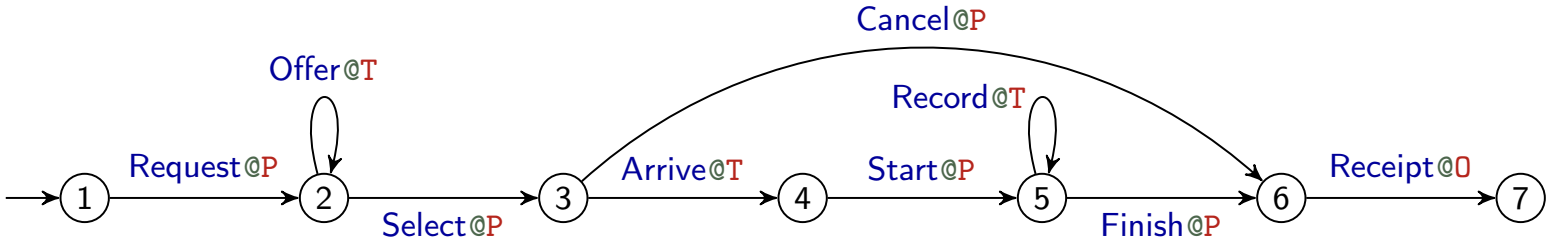
A taxi service

An intuitive auction protocol for a passenger P to get a taxi T:



A taxi service

An intuitive auction protocol for a passenger P to get a taxi T:

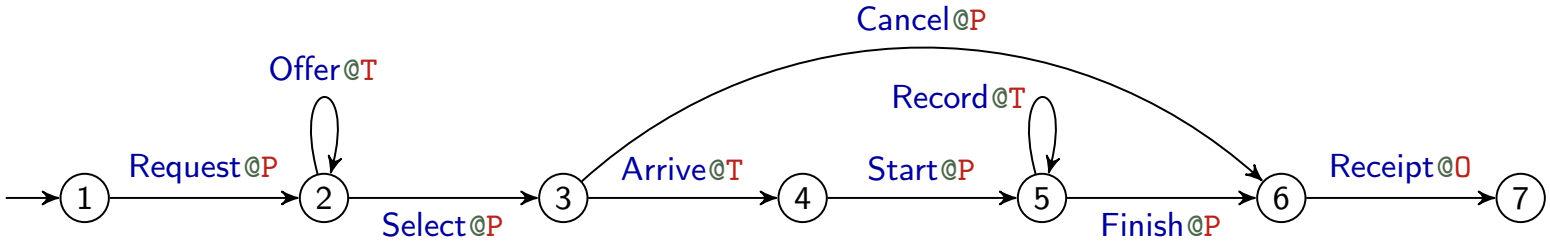


We assume

- one passenger and one office (for simplicity)

A taxi service

An intuitive auction protocol for a passenger P to get a taxi T:

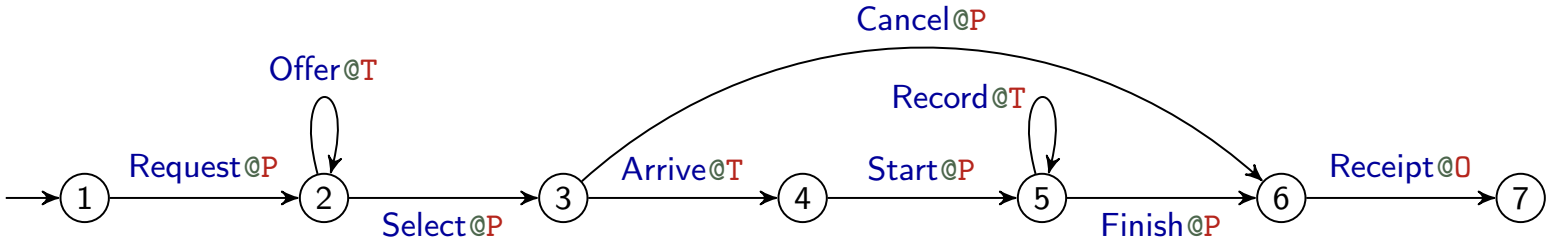


We assume

- one passenger and one office (for simplicity)
- but an arbitrary number of taxis

A taxi service

An intuitive auction protocol for a passenger P to get a taxi T:



We assume

- one passenger and one office (for simplicity)
- but an arbitrary number of taxis
- a receipt is issued by the office O at the end of the ride (if any)

Swarm protocols: global type for local-first applications

An **idealised** specification relying on **synchronous communication**

Fix a set of **roles** ranged over by $\mathtt{R}$ (e.g., $\mathtt{P}$, $\mathtt{T}$, and $\mathtt{O}$)

The syntax of **swarm protocols** is again given co-inductively:

$$\mathtt{G} \overset{\text{co}}{::=} \sum_{i \in I} \mathtt{c}_i@\mathtt{R}_i\langle \mathtt{l}_i \rangle \,.\, \mathtt{G}_i \quad \mid \quad 0 \qquad \text{where } I \text{ is a finite set (of indexes)}$$

An example

A swarm protocol for the taxi scenario:

$$G = \mathsf{Request}@\mathtt{P}\langle \mathsf{Requested} \rangle \,.\, G_{\mathsf{auction}}$$

$$\begin{aligned} G_{\mathsf{auction}} = {} & \mathsf{Offer}@\mathtt{T}\langle \mathsf{Bid} \cdot \mathsf{BidderID} \rangle \,.\, G_{\mathsf{auction}} \\ & + \mathsf{Select}@\mathtt{P}\langle \mathsf{Selected} \cdot \mathsf{PassengerID} \rangle \,.\, G_{\mathsf{choose}} \end{aligned}$$

$$\begin{aligned} G_{\mathsf{choose}} = {} & \mathsf{Arrive}@\mathtt{T}\langle \mathsf{Arrived} \rangle \,.\, \mathsf{Start}@\mathtt{P}\langle \mathsf{Started} \rangle \,.\, G_{\mathsf{ride}} \\ & + \mathsf{Cancel}@\mathtt{P}\langle \mathsf{Cancelled} \rangle \,.\, \mathsf{Receipt}@\mathtt{O}\langle \mathsf{Receipt} \rangle \,.\, 0 \end{aligned}$$

$$\begin{aligned} G_{\mathsf{ride}} = {} & \mathsf{Record}@\mathtt{T}\langle \mathsf{Path} \rangle \,.\, G_{\mathsf{ride}} \\ & + \mathsf{Finish}@\mathtt{P}\langle \mathsf{Finished} \cdot \mathsf{Rating} \rangle \,.\, \mathsf{Receipt}@\mathtt{O}\langle \mathsf{Receipt} \rangle \,.\, 0 \end{aligned}$$

An example

A swarm protocol for the taxi scenario:

$$G = \mathsf{Request}@\mathtt{P}\langle \mathsf{Requested} \rangle \,.\, G_{\mathsf{auction}}$$

$$\begin{aligned} G_{\mathsf{auction}} = {} & \mathsf{Offer}@\mathtt{T}\langle \mathsf{Bid} \cdot \mathsf{BidderID} \rangle \,.\, G_{\mathsf{auction}} \\ & + \mathsf{Select}@\mathtt{P}\langle \mathsf{Selected} \cdot \mathsf{PassengerID} \rangle \,.\, G_{\mathsf{choose}} \end{aligned}$$

Notice the log types in each prefixes

$$\begin{aligned} G_{\mathsf{choose}} = {} & \mathsf{Arrive}@\mathtt{T}\langle \mathsf{Arrived} \rangle \,.\, \mathsf{Start}@\mathtt{P}\langle \mathsf{Started} \rangle \,.\, G_{\mathsf{ride}} \\ & + \mathsf{Cancel}@\mathtt{P}\langle \mathsf{Cancelled} \rangle \,.\, \mathsf{Receipt}@\mathtt{O}\langle \mathsf{Receipt} \rangle \,.\, 0 \end{aligned}$$

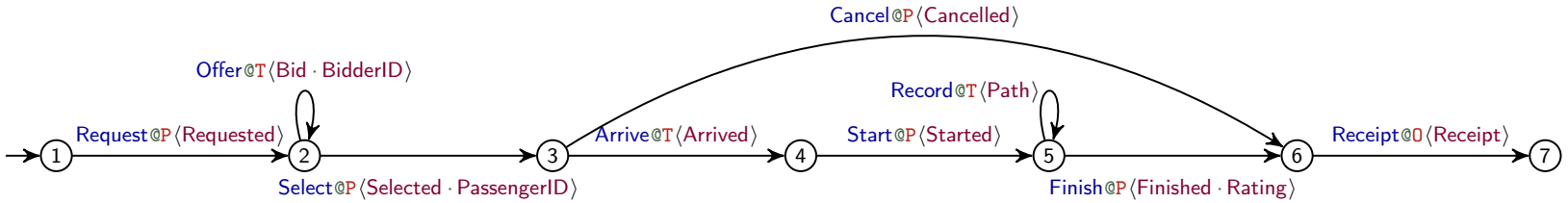
$$\begin{aligned} G_{\mathsf{ride}} = {} & \mathsf{Record}@\mathtt{T}\langle \mathsf{Path} \rangle \,.\, G_{\mathsf{ride}} \\ & + \mathsf{Finish}@\mathtt{P}\langle \mathsf{Finished} \cdot \mathsf{Rating} \rangle \,.\, \mathsf{Receipt}@\mathtt{O}\langle \mathsf{Receipt} \rangle \,.\, 0 \end{aligned}$$

Swarm protocols as FSA

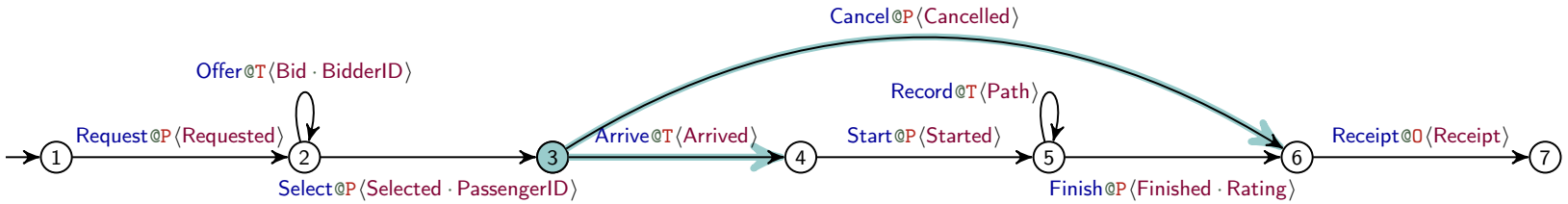
Like for machines, a swarm protocols $\mathsf{G} = \sum_{i \in I} \mathtt{c}_i@\mathtt{R}_i\langle \mathtt{l}_i \rangle \,.\, \mathsf{G}_i$ has an associated FSA:

- the set of states consists of G plus the states in G_i for each $i \in I$
- G is the initial state
- for each $i \in I$, G has a transition to state G_i labelled with $\mathtt{c}_i@\mathtt{R}_i\langle \mathtt{l}_i \rangle$, written $\mathsf{G} \xrightarrow{\mathtt{c}_i \,/\, \mathtt{l}_i} \mathsf{G}_i$

An example


Cancel@P⟨Cancelled⟩
Offer@T⟨Bid · BidderID⟩
Record@T⟨Path⟩
Request@P⟨Requested⟩
Arrive@T⟨Arrived⟩
Start@P⟨Started⟩
Receipt@O⟨Receipt⟩
1
2
3
4
5
6
7
Select@P⟨Selected · PassengerID⟩
Finish@P⟨Finished · Rating⟩

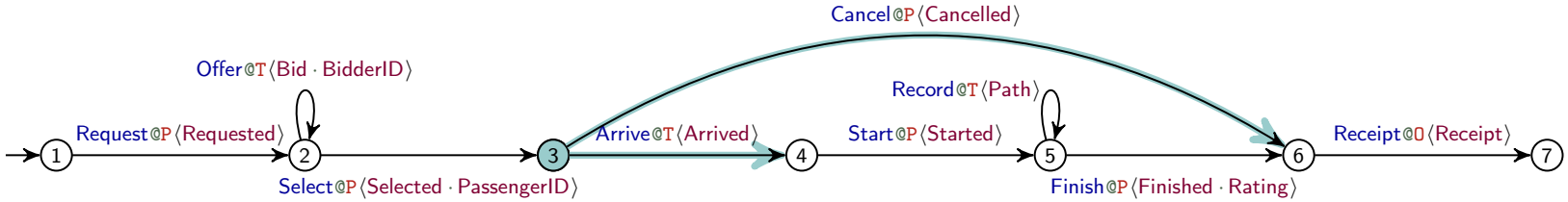
An example



There is a race in state 3!

- the driver of the selected taxi may invoke Arrive
- **while** P loses patience and invokes Cancel

An example



There is a race in state 3!

- the driver of the selected taxi may invoke Arrive
- **while** P loses patience and invokes Cancel

This protocol violates well-formedness conditions typically imposed on behavioural types due to the race in state 3 (because it has two active participants, which is also true of states 2 and 5)

Semantics of swarm protocols

One rule only!

$$\frac{}{(\mathsf{G}, \ell) \xrightarrow{\texttt{c}\,/\,\texttt{l}} (\mathsf{G}, \ell \quad)}\ [\text{G-Cmd}]$$

Semantics of swarm protocols

One rule only!

$$\frac{\delta(\mathsf{G}, \ell) \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathsf{G}'}{(\mathsf{G}, \ell) \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} (\mathsf{G}, \ell \quad)}\ \text{[G-Cmd]}$$

where

$$\delta(\mathsf{G}, \ell) = \begin{cases} \mathsf{G} & \text{if } \ell = \epsilon \\ \delta(\mathsf{G}_1, \ell_2) & \text{if } \mathsf{G} \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathsf{G}' \text{ and } \ell = \ell_1 \cdot \ell_2 \text{ with } \vdash \ell_1 : \mathtt{1} \\ \bot & \text{otherwise} \end{cases}$$

Logs have to be consumed "atomically", hence $\delta(\mathsf{G}, \ell)$ may be undefined

Semantics of swarm protocols

One rule only!

$$\frac{\delta(\mathsf{G}, \ell) \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathsf{G}' \qquad \vdash \ell' : \mathtt{1} \qquad \ell' \text{ log of fresh events}}{(\mathsf{G}, \ell) \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} (\mathsf{G}, \ell \cdot \ell')}\ \text{[G-Cmd]}$$

where

$$\delta(\mathsf{G}, \ell) = \begin{cases} \mathsf{G} & \text{if } \ell = \epsilon \\ \delta(\mathsf{G}_1, \ell_2) & \text{if } \mathsf{G} \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathsf{G}' \text{ and } \ell = \ell_1 \cdot \ell_2 \text{ with } \vdash \ell_1 : \mathtt{1} \\ \bot & \text{otherwise} \end{cases}$$

Logs have to be consumed "atomically", hence $\delta(\mathsf{G}, \ell)$ may be undefined

Semantics of swarm protocols

One rule only!

$$\frac{\delta(\mathsf{G}, \ell) \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathsf{G}' \qquad \vdash \ell' : \mathtt{1} \qquad \ell' \text{ log of fresh events}}{(\mathsf{G}, \ell) \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} (\mathsf{G}, \ell \cdot \ell')}\ \text{[G-Cmd]}$$

where

$$\delta(\mathsf{G}, \ell) = \begin{cases} \mathsf{G} & \text{if } \ell = \epsilon \\ \delta(\mathsf{G}_1, \ell_2) & \text{if } \mathsf{G} \xrightarrow{\mathtt{c}\,/\,\mathtt{1}} \mathsf{G}' \text{ and } \ell = \ell_1 \cdot \ell_2 \text{ with } \vdash \ell_1 : \mathtt{1} \\ \bot & \text{otherwise} \end{cases}$$

Logs have to be consumed "atomically", hence $\delta(\mathsf{G}, \ell)$ may be undefined

From global to local

Transitions of a swarm protocol G are labelled with a role that may invoke the command

From global to local

Transitions of a swarm protocol G are labelled with a role that may invoke the command

Each machine plays one role

From global to local

Transitions of a swarm protocol G are labelled with a role that may invoke the command

Each machine plays one role

Obtain machines by projecting G on each role

From global to local

Transitions of a swarm protocol G are labelled with a role that may invoke the command

Each machine plays one role

Obtain machines by projecting G on each role

First attempt

$$\left(\sum_{i \in I} \mathtt{c}_i @ \mathtt{R}_i \langle \mathtt{l}_i \rangle \,.\, \mathsf{G}_i\right) \downarrow_{\mathtt{R}} = \kappa \cdot [\&_{i \in I}\, \mathtt{l}_i ?\, \mathsf{G}_i \downarrow_{\mathtt{R}}]$$

where $\kappa = \{(\mathtt{c}_i \,/\, \mathtt{l}_i) \mid \mathtt{R}_i = \mathtt{R} \text{ and } i \in I\}$

From global to local

Transitions of a swarm protocol G are labelled with a role that may invoke the command

Each machine plays one role

Obtain machines by projecting G on each role

First attempt

$$\left(\sum_{i \in I} \mathtt{c}_i @ \mathtt{R}_i \langle \mathtt{l}_i \rangle \,.\, \mathsf{G}_i\right) \downarrow_{\mathtt{R}} = \kappa \cdot [\&_{i \in I}\, \mathtt{l}_i ?\, \mathsf{G}_i \downarrow_{\mathtt{R}}]$$

where $\kappa = \{(\mathtt{c}_i \,/\, \mathtt{l}_i) \mid \mathtt{R}_i = \mathtt{R} \text{ and } i \in I\}$

simple, but

- projected machines are large in all but the most trivial cases
- processing **all** events is undesirable: security and efficiency

Another attempt

Let's subscribe to **subscriptions**: maps from roles to sets of event types

In pub-sub, processes subscribe to "topics"

Another attempt

Let's subscribe to **subscriptions**: maps from roles to sets of event types

In pub-sub, processes subscribe to "topics"

Given $\mathsf{G} = \sum_{i \in I} \mathtt{c}_i@\mathtt{R}_i\langle \mathtt{l}_i \rangle \,.\, \mathsf{G}_i$, the **projection of G on a role $\mathtt{R}$ with respect to subscription σ** is

$$\mathsf{G} \downarrow^{\sigma}_{\mathtt{R}} = \kappa \cdot [\&_{j \in J} \, \mathsf{filter}(\mathtt{l}_j, \sigma(\mathtt{R}))? \, \mathsf{G}_j \downarrow^{\sigma}_{\mathtt{R}}] \qquad \text{where}$$

Another attempt

Let's subscribe to **subscriptions**: maps from roles to sets of event types

In pub-sub, processes subscribe to "topics"

Given $\mathsf{G} = \sum_{i \in I} \mathtt{c}_i @ \mathtt{R}_i \langle \mathtt{l}_i \rangle \,.\, \mathsf{G}_i$, the **projection of G on a role $\mathtt{R}$ with respect to subscription σ** is

$$\mathsf{G} \downarrow^{\sigma}_{\mathtt{R}} = \kappa \cdot [\&_{j \in J} \, \mathrm{filter}(\mathtt{l}_j, \sigma(\mathtt{R}))? \, \mathsf{G}_j \downarrow^{\sigma}_{\mathtt{R}}] \qquad \text{where}$$

$$\kappa = \{ \mathtt{c}_i \,/\, \mathtt{l}_i \mid \mathtt{R}_i = \mathtt{R} \text{ and } i \in I \}$$

$$J = \{ i \in I \mid \mathrm{filter}(\mathtt{l}_i, \sigma(\mathtt{R})) \neq \epsilon \}$$

$$\mathrm{filter}(\mathtt{l}, E) = \begin{cases} \epsilon, & \text{if } \mathtt{l} = \epsilon \\ \mathtt{t} \cdot \mathrm{filter}(\mathtt{l}', E) & \text{if } \mathtt{t} \in E \text{ and } \mathtt{l} = \mathtt{t} \cdot \mathtt{l}' \\ \mathrm{filter}(\mathtt{l}, E) & \text{otherwise} \end{cases}$$

Trading consistency for availability has implications:

Causality

Trading consistency for availability has implications:

Propagation of events is non-atomic (cf. rule [Prop])

$\Longrightarrow$ differences in how machines perceive the (state of the) computation

Causality

Fix a subscription σ. For each branch $i \in I$ of $\mathtt{G} = \sum_{i \in I} \mathtt{c}_i@\mathtt{R}_i\langle \mathtt{l}_i \rangle \,.\, \mathtt{G}_i$

Explicit re-enabling $\quad \sigma(\mathtt{R}_i) \cap \mathtt{l}_i \neq \emptyset$

Command causality if $\mathtt{R}$ executes a command in $\mathtt{G}_i$
then $\sigma(\mathtt{R}) \cap \mathtt{l}_i \neq \emptyset$ and $\sigma(\mathtt{R}) \cap \mathtt{l}_i \supseteq \bigcup_{\mathtt{R}' \in_\sigma \mathtt{G}_i} \sigma(\mathtt{R}') \cap \mathtt{l}_i$

Each role explicitly process its emitted events on each of the branches. If $\mathtt{R}$ is supposed to enable a command enabled after $\mathtt{c}_i$ then $\sigma(\mathtt{R})$ contains some event type emitted by $\mathtt{c}_i$

Causality + Determinacy

Trading consistency for availability has implications:

Propagation of events is non-atomic (cf. rule [Prop])

$\Longrightarrow$ different roles may take inconsistent decisions

Causality & Determinacy

Fix a subscription σ. For each branch $i \in I$ of $\mathtt{G} = \sum_{i \in I} \mathtt{c}_i@\mathtt{R}_i\langle \mathtt{l}_i \rangle \,.\, \mathtt{G}_i$

Explicit re-enabling $\quad \sigma(\mathtt{R}_i) \cap \mathtt{l}_i \neq \emptyset$

Command causality $\quad$ if $\mathtt{R}$ executes a command in $\mathtt{G}_i$
then $\sigma(\mathtt{R}) \cap \mathtt{l}_i \neq \emptyset$ and $\sigma(\mathtt{R}) \cap \mathtt{l}_i \supseteq \bigcup_{\mathtt{R}' \in_\sigma \mathtt{G}_i} \sigma(\mathtt{R}') \cap \mathtt{l}_i$

Determinacy $\quad \mathtt{R} \in_\sigma \mathtt{G}_i \implies \mathtt{l}_i[0] \in \sigma(\mathtt{R})$

Causality + Determinacy - Confusion

Trading consistency for availability has implications:

Propagation of events is non-atomic (cf. rule [Prop])

$\Longrightarrow$ branches unambiguously identified and events emitted on eventually discharged branches ignored

Causality & Determinacy & Confusion freeness

Fix a subscription σ. For each branch $i \in I$ of $\mathtt{G} = \sum_{i \in I} \mathtt{c}_i@\mathtt{R}_i\langle \mathtt{l}_i \rangle \,.\, \mathtt{G}_i$

Explicit re-enabling	$\sigma(\mathtt{R}_i) \cap \mathtt{l}_i \neq \emptyset$
Command causality	if $\mathtt{R}$ executes a command in $\mathtt{G}_i$ then $\sigma(\mathtt{R}) \cap \mathtt{l}_i \neq \emptyset$ and $\sigma(\mathtt{R}) \cap \mathtt{l}_i \supseteq \bigcup_{\mathtt{R}' \in_\sigma \mathtt{G}_i} \sigma(\mathtt{R}') \cap \mathtt{l}_i$
Determinacy	$\mathtt{R} \in_\sigma \mathtt{G}_i \implies \mathtt{l}_i[0] \in \sigma(\mathtt{R})$
Confusion freeness	there is a unique subtree $\mathtt{G}'$ of $\mathtt{G}$ emitting $\mathtt{t}$ for each $\mathtt{t}$ starting a log emitted by a command in $\mathtt{G}$

On correctness

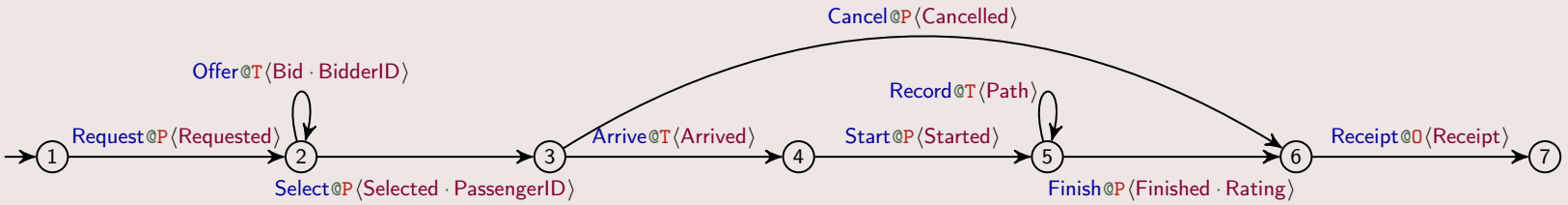
$(\mathtt{S}, \ell)$ faithfully implements G if it produces only logs possibly generated by G

On correctness

$(\mathtt{S}, \ell)$ faithfully implements $\mathtt{G}$ if it produces only logs possibly generated by $\mathtt{G}$

Exercise

Take the swarm $\mathtt{S} = \mathtt{P} \;[]|\; \mathtt{T} \;[]|\; \mathtt{O} \;[]|\; \mathtt{T} \;[]$ implementing our taxi protocol



Check that $\mathtt{S}$ generates the log

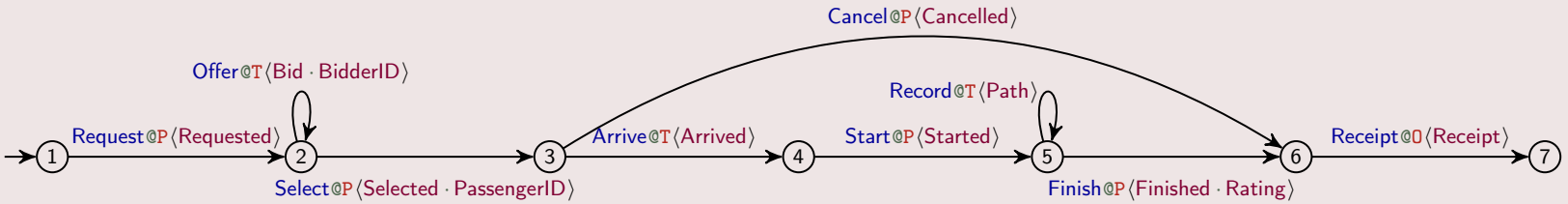
$$\ell_{\text{auc}} = \mathit{requested} \cdot \mathit{bid} \cdot \mathit{bidderID} \cdot \mathit{selected} \cdot \mathit{bid} \cdot \mathit{bidderID} \cdot \mathit{passengerID}$$

On correctness

$(\mathtt{S}, \ell)$ faithfully implements $\mathtt{G}$ if it produces only logs possibly generated by $\mathtt{G}$

Exercise

Take the swarm $\mathtt{S} = \mathtt{P} \;[]|\; \mathtt{T} \;[]|\; \mathtt{O} \;[]|\; \mathtt{T} \;[]$ implementing our taxi protocol



Check that $\mathtt{S}$ generates the log

$$\ell_{\text{auc}} = \mathit{requested} \cdot \mathit{bid} \cdot \mathit{bidderID} \cdot \mathit{selected} \cdot \mathit{bid} \cdot \mathit{bidderID} \cdot \mathit{passengerID}$$

Too strong a requirement!

Let's consider only "good enough" logs, i.e., those typeable with $\mathtt{G}$'s log types

References I

[1] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*.
MIT Press, Cambridge, MA, USA, 1986.

[2] Daniel Brand and Pitro Zafiropulo. On Communicating Finite-State Machines.
JACM, 30(2):323–342, 1983.

[3] Pierre-Malo Deniélou and Nobuko Yoshida. Multiparty session types meet communicating automata.
In *ESOP 2012*, pages 194–213, 2012.

[4] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process.
J. ACM, 32(2):374–382, apr 1985.

[5] Roberto Guanciale and Emilio Tuosto. An abstract semantics of the global view of choreographies.
In *Proceedings 9th Interaction and Concurrency Experience, ICE 2016, Heraklion, Greece, 8-9 June 2016.*, pages 67–82, 2016.

[6] Roberto Guanciale and Emilio Tuosto. Realisability of pomsets.
Journal of Logic and Algebraic Methods in Programming, 108:69–89, 2019.

[7] Carl Hewitt, Peter Boehler Bishop, and Richard Steiger. A Universal Modular ACTOR Formalism for Artificial Intelligence.
In Nils J. Nilsson, editor, *Proceedings of the 3rd International Joint Conference on Artificial Intelligence. Standford, CA, USA, August 20-23, 1973*, pages 235–245. William Kaufmann, 1973.

[8] Nickolas Kavantzas, Davide Burdett, Gregory Ritzinger, Tony Fletcher, and Yves Lafon. Web services choreography description language version 1.0.
http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217. Working Draft 17 December 2004.

References II

[9] Roland Kuhn, Hernán C. Melgratti, and Emilio Tuosto. Behavioural types for local-first software. In Karim Ali and Guido Salvaneschi, editors, *37th European Conference on Object-Oriented Programming, ECOOP 2023, July 17-21, 2023, Seattle, Washington, United States*, volume 263 of *LIPIcs*, pages 15:1–15:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[10] Julien Lange, Emilio Tuosto, and Nobuko Yoshida. From Communicating Machines to Graphical Choreographies. In *POPL15*, pages 221–232, 2015.

[11] Emilio Tuosto and Roberto Guanciale. Semantics of global view of choreographies. *Journal of Logic and Algebraic Methods in Programming*, 95:17–40, 2018. Revised and extended version of [5]. available at http://www.cs.le.ac.uk/people/et52/jlamp-with-proofs.pdf.