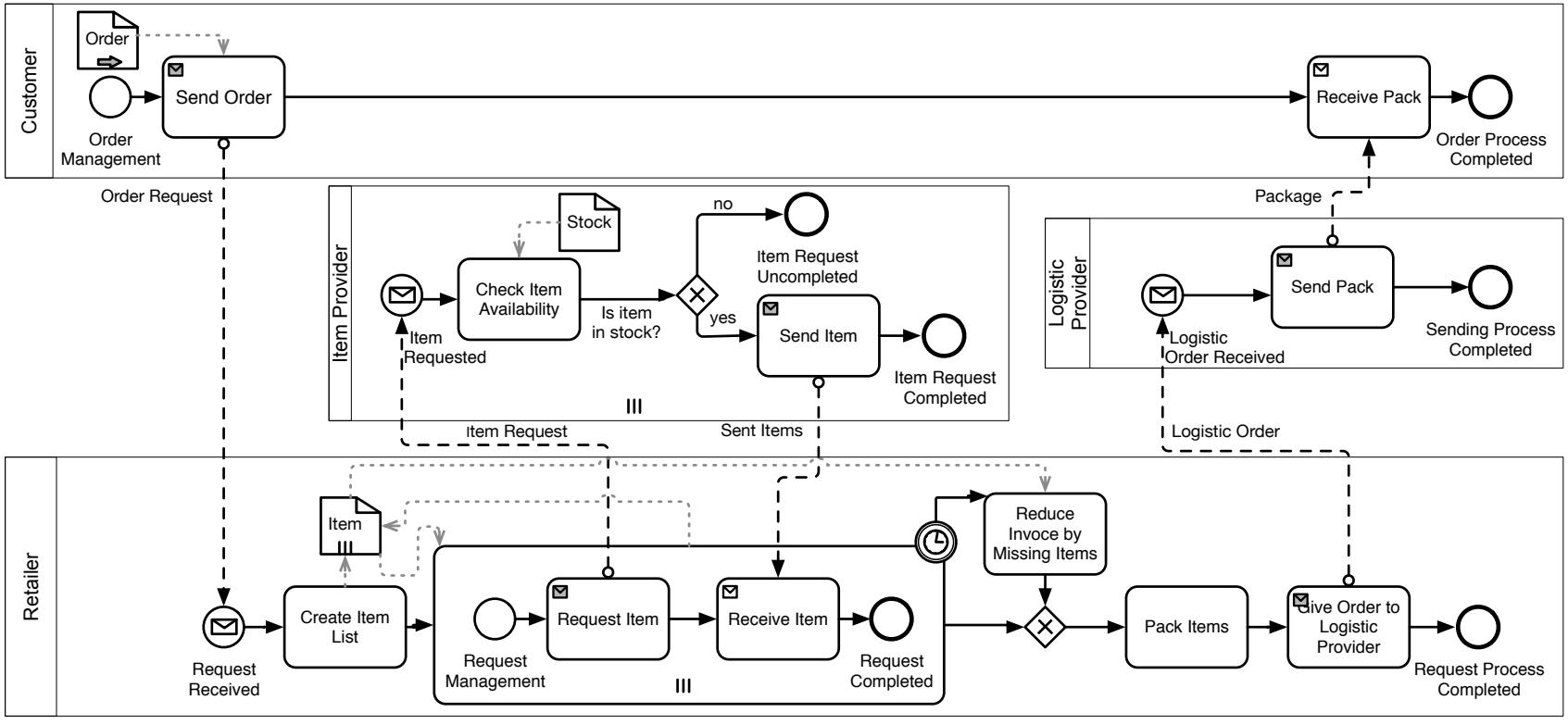
Formalising BPMN Service Interaction Patterns

Barbara Re

December 10, 2018

Motivations

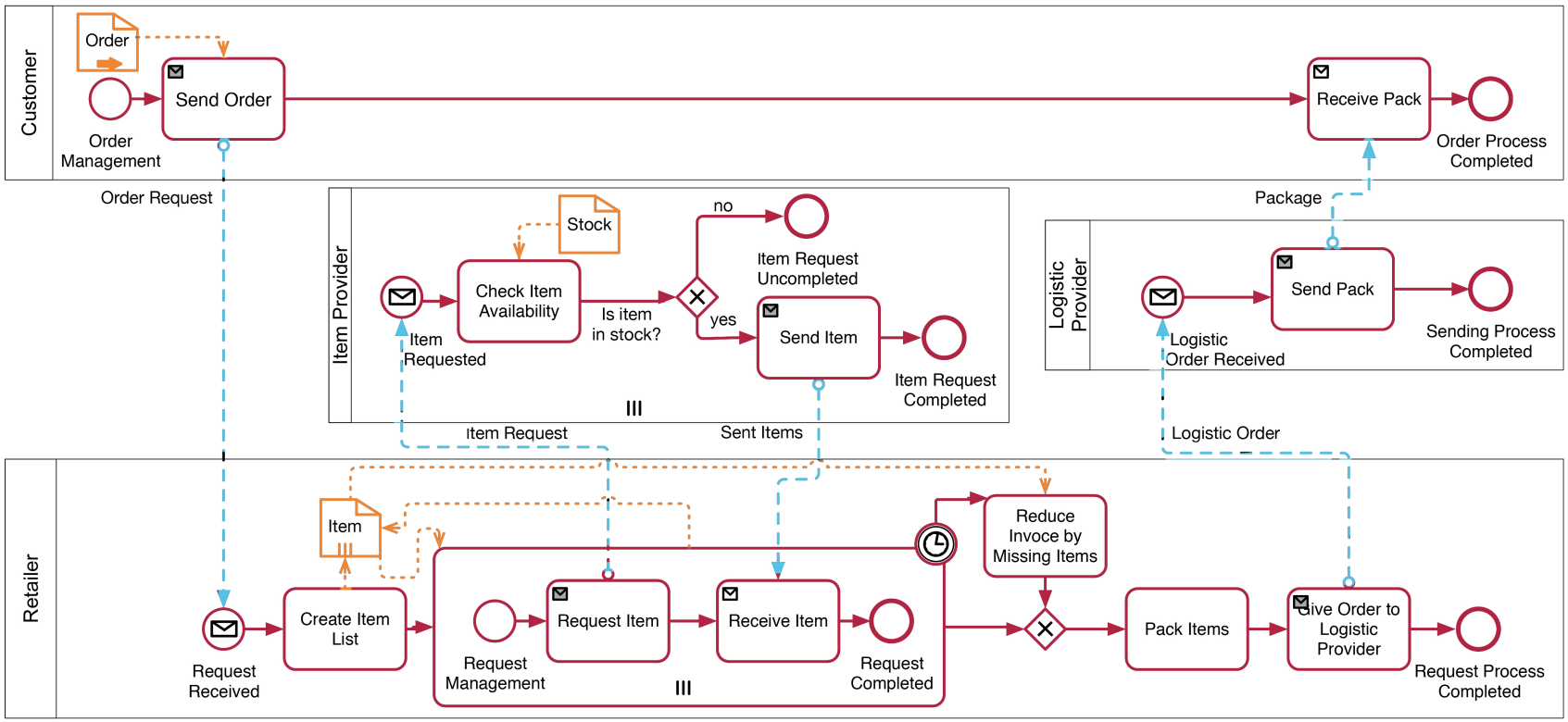
Interactions are first-class citizens in BPMN collaborations



Organisations often **interact with other parties**

Motivations

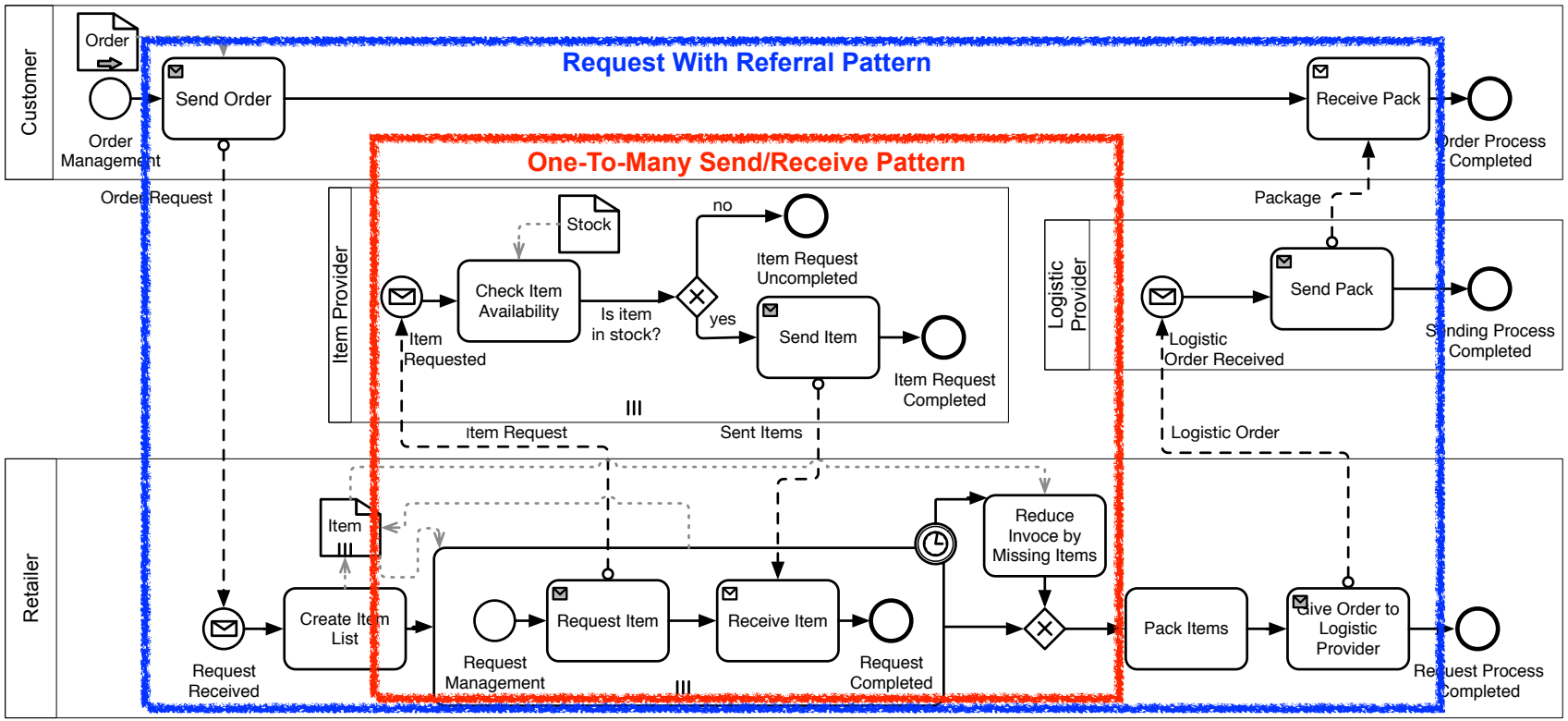
Interactions are first-class citizens in BPMN collaborations



BPMN provides an intuitive description of the process behaviour, distinguishing **control**, **message** and **data** flow

Motivations

Interactions are first-class citizens in BPMN collaborations



Researchers spent a lot of efforts to identify the most common interaction scenarios: the **Service Interaction Patterns**

Motivations

Service interaction patterns[1] are textually provided

- A **visualisation** as well as a **formalisation** is crucial to precisely render the message exchanges between participants
- Without a clear understanding of the interplay of control, message and data flow **different interpretations** may easily arise

Service interactions range from a single message exchange, to multiple instance participants communications and routing behaviours

[1] Barros A., Dumas M., ter Hofstede A.H.M. (2005) Service Interaction Patterns. In: Business Process Management. BPM 2005. LNCS, vol 3649. Springer.

Main Focus

Formalise the execution semantics of service interaction patterns specified in BPMN

- A **formal semantics** for BPMN, given in **SOS** style: that associates a LTS to each model
- A BPMN collaboration model and its formalisation in terms of **LTS transitions** for each interaction pattern
- The tool MIDA for **pattern animations**, that faithfully implements the semantics

A direct semantics for BPMN multi-instance collaborations

Formal Semantics

To simplify the formal treatment we resort to a **textual representation**

$$
\begin{aligned}
C &::= \mathsf{pool}(\mathsf{p}, P) \mid \mathsf{miPool}(\mathsf{p}, P) \mid C_1 \parallel C_2 \\
P &::= \mathsf{start}(\mathsf{e}_{enb}, \mathsf{e}_o) \mid \mathsf{startRcv}(\mathsf{m}:\tilde{\mathsf{t}}, \mathsf{e}_o) \mid \mathsf{end}(\mathsf{e}_i) \mid \mathsf{endSnd}(\mathsf{e}_i, \mathsf{m}:\tilde{\mathsf{exp}}) \mid \mathsf{terminate}(\mathsf{e}_i) \\
&\quad \mid \mathsf{andSplit}(\mathsf{e}_i, E_o) \mid \mathsf{xorSplit}(\mathsf{e}_i, G) \mid \mathsf{andJoin}(E_i, \mathsf{e}_o) \mid \mathsf{xorJoin}(E_i, \mathsf{e}_o) \\
&\quad \mid \mathsf{eventBased}(\mathsf{e}_i, (\mathsf{m}_1:\tilde{\mathsf{t}}_1, \mathsf{e}_{o1}), \ldots, (\mathsf{m}_h:\tilde{\mathsf{t}}_h, \mathsf{e}_{oh})) \\
&\quad \mid \mathsf{task}(\mathsf{e}_i, \mathsf{exp}, A, \mathsf{e}_o) \mid \mathsf{taskRcv}(\mathsf{e}_i, \mathsf{exp}, A, \mathsf{m}:\tilde{\mathsf{t}}, \mathsf{e}_o) \mid \mathsf{taskSnd}(\mathsf{e}_i, \mathsf{exp}, A, \mathsf{m}:\tilde{\mathsf{exp}}, \mathsf{e}_o) \\
&\quad \mid \mathsf{interRcv}(\mathsf{e}_i, \mathsf{m}:\tilde{\mathsf{t}}, \mathsf{e}_o) \mid \mathsf{interSnd}(\mathsf{e}_i, \mathsf{m}:\tilde{\mathsf{exp}}, \mathsf{e}_o) \mid P_1 \parallel P_2 \\
A &::= \epsilon \mid \mathsf{d.f} ::= \mathsf{exp}, A
\end{aligned}
$$

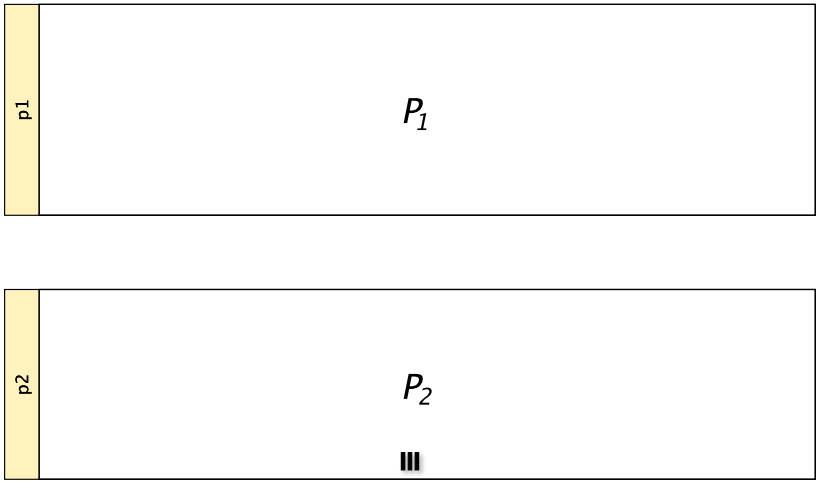
Formal Semantics

To simplify the formal treatment we resort to a **textual representation**

$C ::= \mathrm{pool}(\mathrm{p}, P) \mid \mathrm{miPool}(\mathrm{p}, P) \mid C_1 \parallel C_2$

$P ::= \mathrm{start}(\mathrm{e}_{enb}, \mathrm{e}_o) \mid \mathrm{startRcv}(\mathrm{m} : \tilde{\mathrm{t}}, \mathrm{e}_o) \mid \mathrm{end}(\mathrm{e}_i) \mid \mathrm{endSnd}(\mathrm{e}_i, \mathrm{m} : \tilde{\mathrm{exp}}) \mid \mathrm{terminate}(\mathrm{e}_i)$
$\mid \mathrm{andSplit}(\mathrm{e}_i, E_o) \mid \mathrm{xorSplit}(\mathrm{e}_i, G) \mid \mathrm{andJoin}(E_i, \mathrm{e}_o) \mid \mathrm{xorJoin}(E_i, \mathrm{e}_o)$
$\mid \mathrm{eventBased}(\mathrm{e}_i, (\mathrm{m}_1 : \tilde{\mathrm{t}}_1, \mathrm{e}_{o1}), \ldots, (\mathrm{m}_h : \tilde{\mathrm{t}}_h, \mathrm{e}_{oh}))$
$\mid \mathrm{task}(\mathrm{e}_i, \mathrm{exp}, A, \mathrm{e}_o) \mid \mathrm{taskRcv}(\mathrm{e}_i, \mathrm{exp}, A, \mathrm{m} : \tilde{\mathrm{t}}, \mathrm{e}_o) \mid \mathrm{taskSnd}(\mathrm{e}_i, \mathrm{exp}, A, \mathrm{m} : \tilde{\mathrm{exp}}, \mathrm{e}_o)$
$\mid \mathrm{interRcv}(\mathrm{e}_i, \mathrm{m} : \tilde{\mathrm{t}}, \mathrm{e}_o) \mid \mathrm{interSnd}(\mathrm{e}_i, \mathrm{m} : \tilde{\mathrm{exp}}, \mathrm{e}_o) \mid P_1 \parallel P_2$

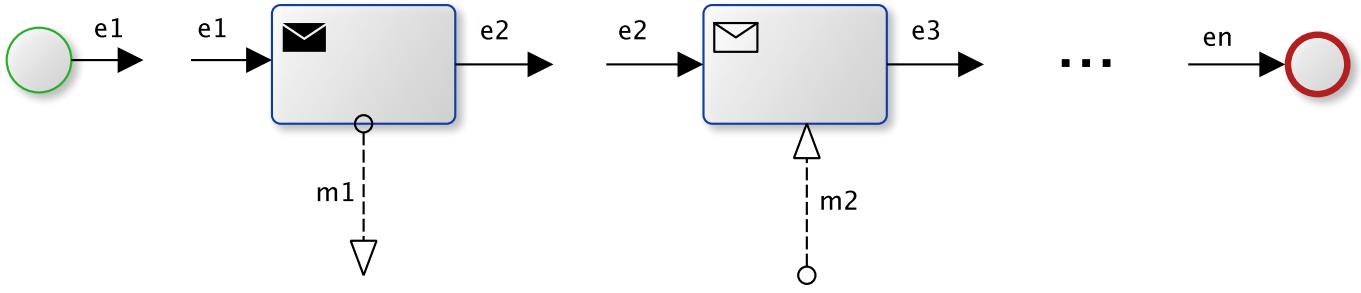
$A ::= \epsilon \mid \mathrm{d.f} ::= \mathrm{exp}, A$



Formal Semantics

To simplify the formal treatment we resort to a **textual representation**

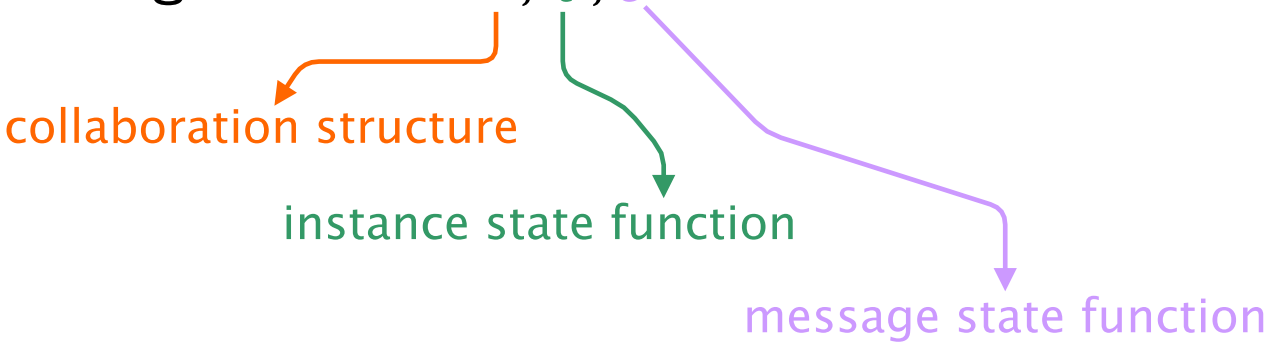
$$
\begin{aligned}
C &::= \mathrm{pool}(\mathsf{p}, P) \mid \mathrm{miPool}(\mathsf{p}, P) \mid C_1 \parallel C_2 \\
P &::= \mathrm{start}(\mathsf{e}_{enb}, \mathsf{e}_o) \mid \mathrm{startRcv}(\mathsf{m}:\tilde{\mathsf{t}}, \mathsf{e}_o) \mid \mathrm{end}(\mathsf{e}_i) \mid \mathrm{endSnd}(\mathsf{e}_i, \mathsf{m}:\tilde{\mathsf{exp}}) \mid \mathrm{terminate}(\mathsf{e}_i) \\
&\quad \mid \mathrm{andSplit}(\mathsf{e}_i, E_o) \mid \mathrm{xorSplit}(\mathsf{e}_i, G) \mid \mathrm{andJoin}(E_i, \mathsf{e}_o) \mid \mathrm{xorJoin}(E_i, \mathsf{e}_o) \\
&\quad \mid \mathrm{eventBased}(\mathsf{e}_i, (\mathsf{m}_1:\tilde{\mathsf{t}}_1, \mathsf{e}_{o1}), \ldots, (\mathsf{m}_h:\tilde{\mathsf{t}}_h, \mathsf{e}_{oh})) \\
&\quad \mid \mathrm{task}(\mathsf{e}_i, \mathrm{exp}, A, \mathsf{e}_o) \mid \mathrm{taskRcv}(\mathsf{e}_i, \mathrm{exp}, A, \mathsf{m}:\tilde{\mathsf{t}}, \mathsf{e}_o) \mid \mathrm{taskSnd}(\mathsf{e}_i, \mathrm{exp}, A, \mathsf{m}:\tilde{\mathsf{exp}}, \mathsf{e}_o) \\
&\quad \mid \mathrm{interRcv}(\mathsf{e}_i, \mathsf{m}:\tilde{\mathsf{t}}, \mathsf{e}_o) \mid \mathrm{interSnd}(\mathsf{e}_i, \mathsf{m}:\tilde{\mathsf{exp}}, \mathsf{e}_o) \mid P_1 \parallel P_2 \\
A &::= \epsilon \mid \mathsf{d.f} ::= \mathrm{exp}, A
\end{aligned}
$$



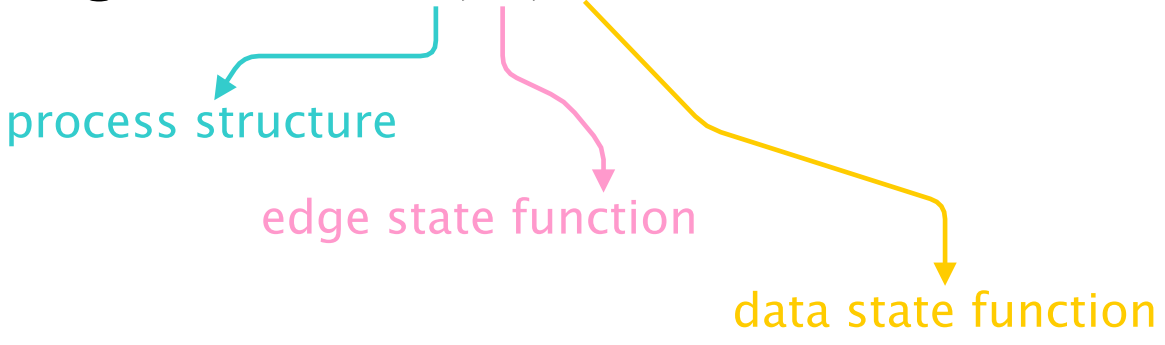
We support models with **arbitrary topology**

Formal Semantics

- The textual representation represents the mere structure of processes and collaborations
- The semantics is given in terms of **stateful** descriptions, called **collaboration configurations** $< C, \iota, \delta >$



and **process configurations** $< P, \sigma, \alpha >$



Formal Semantics: Operational Rules

- **Operational Semantics**: labelled transitions defined by rules of the form

$$\frac{premise_1 \quad \ldots \quad premise_n}{conclusion} \; conditions$$

- **Compositional** approach
  - Collaboration semantics is given in terms of the semantics of its pools
  - Pool semantics is given in terms of the semantics of its process
  - Process semantics is given in terms of the semantics of its elements
  - Each process element has its semantic rules

$$\langle \mathsf{taskSnd}(\mathsf{e}_i, \mathsf{exp}', A, \mathsf{m{:}e\tilde{x}p}, \mathsf{e}_o), \sigma, \alpha \rangle \xrightarrow{!\mathsf{m} : \tilde{\mathsf{v}}} \langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), \alpha' \rangle \quad \begin{array}{l} \sigma(\mathsf{e}_i) > 0, \\ eval(\mathsf{exp}', \alpha, true), \\ upd(\alpha, A, \alpha'), \\ eval(\mathsf{e\tilde{x}p}, \alpha, \tilde{\mathsf{v}}) \end{array}$$

Formal Semantics: Operational Rules - Process

$$\langle \mathsf{start}(\mathsf{e}_{enb}, \mathsf{e}_o), \sigma, \alpha \rangle \xrightarrow{\epsilon} inc(reset(\sigma, \mathsf{e}_{enb}), \mathsf{e}_o) \quad \sigma(\mathsf{e}_{enb}) > 0 \quad (P\text{-}Start)$$

$$\langle \mathsf{end}(\mathsf{e}_i), \sigma, \alpha \rangle \xrightarrow{\epsilon} dec(\sigma, \mathsf{e}_i) \quad \sigma(\mathsf{e}_i) > 0 \quad (P\text{-}End)$$

$$\langle \mathsf{startRcv}(\mathsf{m}:\tilde{\mathsf{t}}, \mathsf{e}_o), \sigma, \alpha \rangle \xrightarrow{new\ \mathsf{m}:\tilde{\mathsf{et}}} inc(\sigma, \mathsf{e}_o) \quad eval(\tilde{\mathsf{t}}, \alpha, \tilde{\mathsf{et}}) \quad (P\text{-}StartRcv)$$

$$\langle \mathsf{xorSplit}(\mathsf{e}_i, \{(\mathsf{e}, \mathsf{exp})\} \cup G), \sigma, \alpha \rangle \xrightarrow{\epsilon} inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}) \quad \sigma(\mathsf{e}_i) > 0,\ eval(\mathsf{exp}, \alpha, true) \quad (P\text{-}XorSplit_1)$$

$$\langle \mathsf{xorSplit}(\mathsf{e}_i, \{(\mathsf{e}, \mathsf{default})\} \cup G), \sigma, \alpha \rangle \xrightarrow{\epsilon} inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}) \quad \sigma(\mathsf{e}_i) > 0,\ \forall (\mathsf{e}_j, \mathsf{exp}_j) \in G\ .\ eval(\mathsf{exp}_j, \alpha, false) \quad (P\text{-}XorSplit_2)$$

$$\langle \mathsf{xorJoin}(\{\mathsf{e}\} \cup E_i, \mathsf{e}_o), \sigma, \alpha \rangle \xrightarrow{\epsilon} inc(dec(\sigma, \mathsf{e}), \mathsf{e}_o) \quad \sigma(\mathsf{e}) > 0 \quad (P\text{-}XorJoin)$$

$$\langle \mathsf{task}(\mathsf{e}_i, \mathsf{exp}, A, \mathsf{e}_o), \sigma, \alpha \rangle \xrightarrow{\epsilon} \langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), \alpha' \rangle \quad \sigma(\mathsf{e}_i) > 0,\ eval(\mathsf{exp}, \alpha, true),\ upd(\alpha, A, \alpha') \quad (P\text{-}Task)$$

$$\langle \mathsf{taskRcv}(\mathsf{e}_i, \mathsf{exp}, A, \mathsf{m}:\tilde{\mathsf{t}}, \mathsf{e}_o), \sigma, \alpha \rangle \xrightarrow{?\mathsf{m}:\tilde{\mathsf{et}}, A} inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o) \quad \sigma(\mathsf{e}_i) > 0,\ eval(\mathsf{exp}, \alpha, true),\ eval(\tilde{\mathsf{t}}, \alpha, \tilde{\mathsf{et}}) \quad (P\text{-}TaskRcv)$$

$$\langle \mathsf{taskSnd}(\mathsf{e}_i, \mathsf{exp}', A, \mathsf{m}:\tilde{\mathsf{exp}}, \mathsf{e}_o), \sigma, \alpha \rangle \xrightarrow{!\mathsf{m}:\tilde{\mathsf{v}}} \langle inc(dec(\sigma, \mathsf{e}_i), \mathsf{e}_o), \alpha' \rangle \quad \sigma(\mathsf{e}_i) > 0,\ eval(\mathsf{exp}', \alpha, true),\ upd(\alpha, A, \alpha'),\ eval(\tilde{\mathsf{exp}}, \alpha, \tilde{\mathsf{v}}) \quad (P\text{-}TaskSnd)$$

$$\frac{\langle P_1, \sigma, \alpha \rangle \xrightarrow{\ell} \langle \sigma', \alpha' \rangle}{\langle P_1 \parallel P_2, \sigma, \alpha \rangle \xrightarrow{\ell} \langle \sigma', \alpha' \rangle} \quad \ell \neq kill \quad (P\text{-}Int_1)$$

Formal Semantics: Operational Rules - Collaboration

$$\frac{\langle P, \sigma_0, \alpha_0\rangle \xrightarrow{new\ \mathsf{m}:\tilde{\mathsf{et}}} \langle \sigma', \alpha'\rangle \quad \tilde{\mathsf{v}} \in \delta(\mathsf{m}) \quad match(\tilde{\mathsf{et}}, \tilde{\mathsf{v}}) = A \quad upd(\alpha', A, \alpha'')}{\langle \mathsf{miPool}(\mathsf{p}, P), \iota, \delta\rangle \xrightarrow{new\ \mathsf{m}:\tilde{\mathsf{v}}} \langle newI(\iota, \mathsf{p}, \sigma', \alpha''), rm(\delta, \mathsf{m}, \tilde{\mathsf{v}})\rangle} (C\text{-}CreateMi)$$

$$\frac{\iota(\mathsf{p}) = \{\langle \sigma, \alpha\rangle\} + I \qquad \langle P, \sigma, \alpha\rangle \xrightarrow{\tau} \langle \sigma', \alpha'\rangle}{\langle \mathsf{miPool}(\mathsf{p}, P), \iota, \delta\rangle \xrightarrow{\tau} \langle updI(\iota, \mathsf{p}, \{\langle \sigma', \alpha'\rangle\} + I), \delta\rangle} (C\text{-}InternalMi)$$

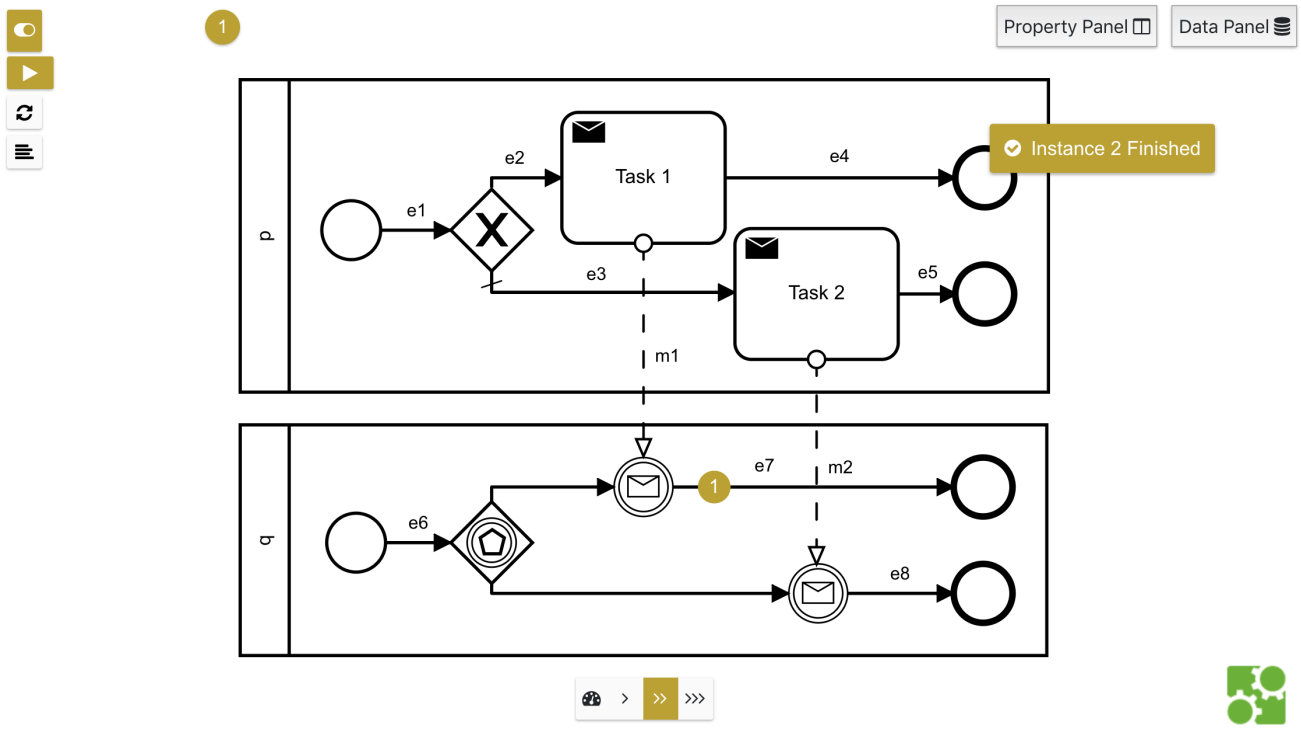
$$\frac{\iota(\mathsf{p}) = \{\langle \sigma, \alpha\rangle\} + I \qquad \langle P, \sigma, \alpha\rangle \xrightarrow{?\mathsf{m}:\tilde{\mathsf{et}}, A} \langle \sigma', \alpha'\rangle \quad \tilde{\mathsf{v}} \in \delta(\mathsf{m}) \quad match(\tilde{\mathsf{et}}, \tilde{\mathsf{v}}) = A' \quad upd(\alpha', (A', A), \alpha'')}{\langle \mathsf{miPool}(\mathsf{p}, P), \iota, \delta\rangle \xrightarrow{?\mathsf{m}:\tilde{\mathsf{v}}} \langle updI(\iota, \mathsf{p}, \{\langle \sigma', \alpha''\rangle\} + I), rm(\delta, \mathsf{m}, \tilde{\mathsf{v}})\rangle} (C\text{-}ReceiveMi)$$

$$\frac{\iota(\mathsf{p}) = \{\langle \sigma, \alpha\rangle\} + I \qquad \langle P, \sigma, \alpha\rangle \xrightarrow{!\mathsf{m}:\tilde{\mathsf{v}}} \langle \sigma', \alpha'\rangle}{\langle \mathsf{miPool}(\mathsf{p}, P), \iota, \delta\rangle \xrightarrow{!\mathsf{m}:\tilde{\mathsf{v}}} \langle updI(\iota, \mathsf{p}, \{\langle \sigma', \alpha'\rangle\} + I), add(\delta, \mathsf{m}, \tilde{\mathsf{v}})\rangle} (C\text{-}DeliverMi)$$

$$\frac{\langle C_1, \iota, \delta\rangle \xrightarrow{l} \langle \iota', \delta'\rangle}{\langle C_1 \parallel C_2, \iota, \delta\rangle \xrightarrow{l} \langle \iota', \delta'\rangle} (C\text{-}Int_1)$$

Process animation in MIDA

MIDA is a free web application for animating BPMN collaboration with multiple instances, data and messages



We designed all the formalised pattern with MIDA providing their animation here

http://pros.unicam.it/service-interaction-patterns/

Patterns Formalisation

Patterns Formalisation

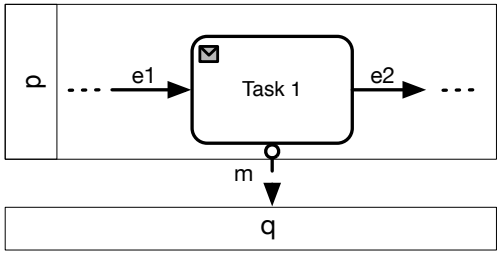
BPMN supports a subset of service patterns:

- Send
- Racing Incoming
- One To Many Send/Receive
- Receive
- One To Many Send
- Request With Referral
- Send/Receive
- One From Many Receive
- Request With Relay

- A textual specification of the corresponding **BPMN fragment**
- The formal semantics in terms of **operational rules** applied for execute the pattern
- The fragment **animation** in MIDA

Send Pattern

Informal Description: A party sends a message to another one.



Textual Specification:

$$C \quad = \quad \text{pool}(\text{p}, P) \parallel \text{pool}(\text{q}, Q), \textit{where}$$

$P = \text{taskSnd}(\text{e1}, \text{exp}_1, \epsilon, \text{m} : \tilde{\text{exp}}_2, \text{e2}) \parallel P'$

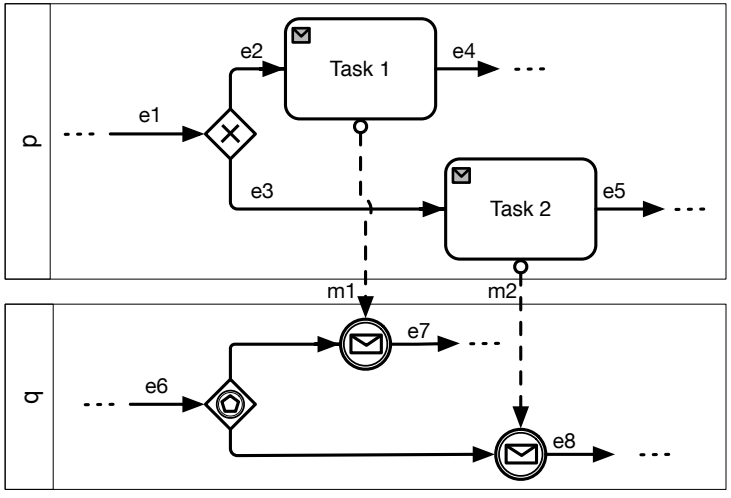
Formal Semantics: *p* execute Task 1 applying the rule *P-TaskSnd*

$$\langle \text{taskSnd}(\text{e}_i, \text{exp}', A, \text{m} : \tilde{\text{exp}}, \text{e}_o), \sigma, \alpha \rangle \xrightarrow{!\text{m} : \tilde{\text{v}}} \langle \textit{inc}(\textit{dec}(\sigma, \text{e}_i), \text{e}_o), \alpha' \rangle$$

$$\sigma(\text{e}_i) > 0,\ \textit{eval}(\text{exp}', \alpha, \textit{true}),\ \textit{upd}(\alpha, A, \alpha'),\ \textit{eval}(\tilde{\text{exp}}, \alpha, \tilde{\text{v}})$$

Racing Incoming Message Pattern

Informal Description: A party expects to receive one among a set of messages.

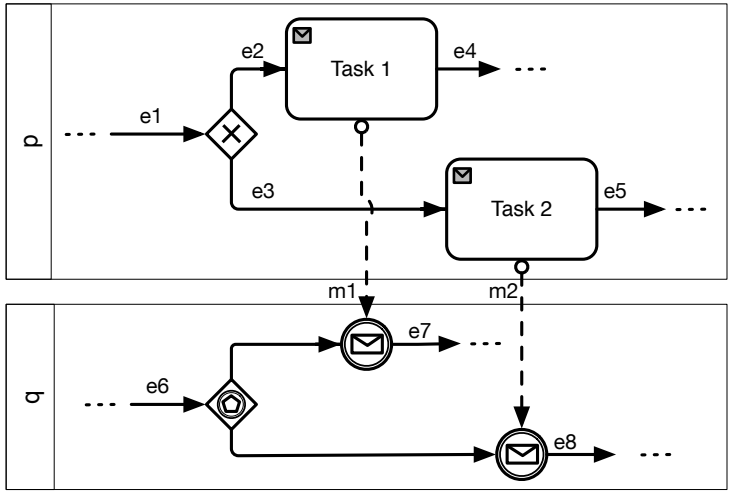


Textual Specification:

$$C = \text{pool}(\text{p}, P) \parallel \text{pool}(\text{q}, Q), \text{ where}$$

Racing Incoming Message Pattern

Informal Description: A party expects to receive one among a set of messages.

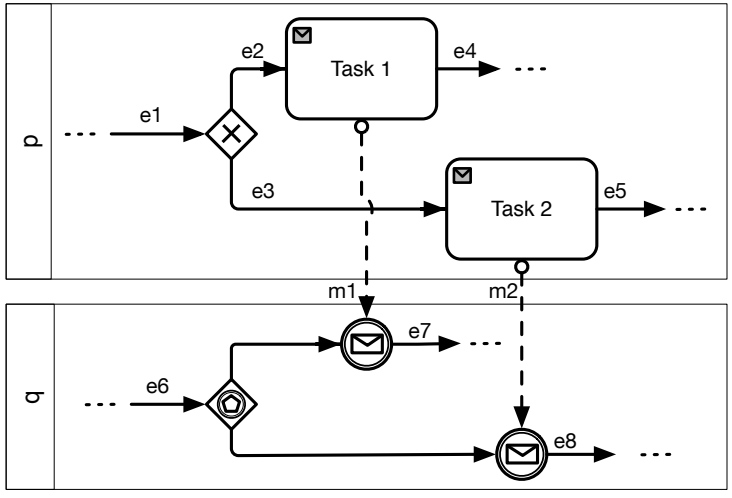


Textual Specification:

$$
\begin{aligned}
P &= \text{xorSplit}(\text{e1}, \{(\text{e2}, \text{exp}_1), (\text{e3}, \text{exp}_2)\}) \parallel \text{taskSnd}(\text{e2}, \text{exp}_3, \epsilon, \text{m1} : \tilde{\text{exp}}_4, \text{e4}) \parallel \\
&\quad \text{taskSnd}(\text{e3}, \text{exp}_5, \epsilon, \text{m2} : \tilde{\text{exp}}_6, \text{e5}) \parallel P' \\
Q &= \text{eventBased}(\text{e6}, (\text{m1} : \tilde{\text{t}}_1, \text{e7}), (\text{m2} : \tilde{\text{t}}_2, \text{e8})) \parallel Q'
\end{aligned}
$$

Racing Incoming Message Pattern

Informal Description: A party expects to receive one among a set of messages.

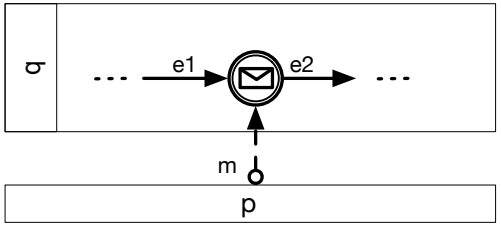


Formal Semantics: process q apply rule P-EventG

$$\langle \text{eventBased}(e_i, (m_1 : \tilde{t}_1, e_{o1}), \ldots, (m_h : \tilde{t}_h, e_{oh})), \sigma, \alpha \rangle \xrightarrow{?m_j : \tilde{et}_j, \epsilon} inc(dec(\sigma, e_i), e_{oj}) \quad \sigma(e_i) > 0, 1 \leqslant j \leqslant h \; eval(\tilde{t}_j, \alpha, \tilde{et}_j)$$

Receive Pattern

Informal Description: A party receives a message from another party.



Textual Specification:

$$C \quad = \quad \text{pool}(\text{p}, P) \parallel \text{pool}(\text{q}, Q), \textit{where}$$

$Q = \text{interRcv}(\text{e1}, \text{m} : \tilde{\text{t}}, \text{e2}) \parallel Q'$

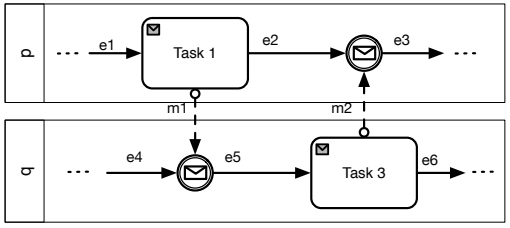
Formal Semantics: *q* receive message *m* applying the rule *P-InterRcv*

$$\langle \text{interRcv}(\text{e}_i, \text{m} : \tilde{\text{t}}, \text{e}_o), \sigma, \alpha \rangle \xrightarrow{?\text{m} : \tilde{\text{et}}, \epsilon} \langle \textit{inc}(\textit{dec}(\sigma, \text{e}_i), \text{e}_o), \alpha \rangle$$

$$\sigma(\text{e}_i) > 0, \quad \textit{eval}(\tilde{\text{t}}, \alpha, \tilde{\text{et}})$$

Send/Receive Pattern

Informal Description: Two parties, p and q, engage in two causally related interactions.



Textual Specification:

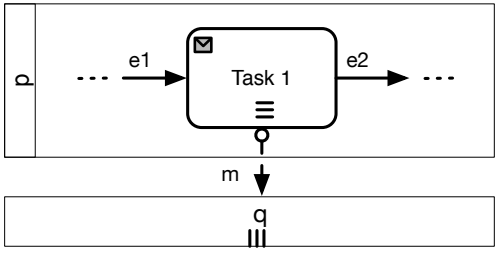
$$C \quad = \quad \text{pool}(\text{p}, P) \parallel \text{pool}(\text{q}, Q), \textit{where}$$

$P = \text{taskSnd}(\text{e1}, \text{exp}_1, \epsilon, \text{m1} : \tilde{\text{exp}}_2, \text{e2}) \parallel \text{interRcv}(\text{e2}, \text{m2} : \tilde{\text{t}_1}, \text{e3}) \parallel P'$
$Q = \text{interRcv}(\text{e4}, \text{m1} : \tilde{\text{t}_2}, \text{e5}) \parallel \text{taskSnd}(\text{e5}, \text{exp}_3, \epsilon, \text{m2} : \tilde{\text{exp}}_4, \text{e6}) \parallel Q'$

Formal Semantics: *p* sends message *m*1 to *q* that reply with *m*2 applying the rules *C-Deliver* and *P-InterRcv*

One-To-Many Send Pattern

Informal Description: A party sends messages to several parties.



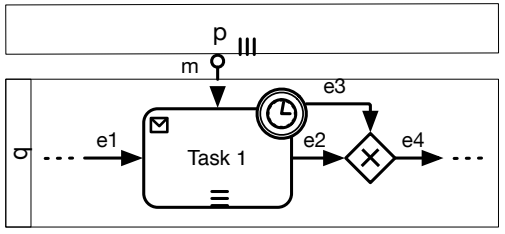
Textual Specification:

$$C = \mathsf{pool}(\mathsf{p}, P) \parallel \mathsf{miPool}(\mathsf{q}, Q), \text{ where}$$

$$\begin{aligned} P \quad = \quad & \mathsf{xorJoin}(\{\mathsf{e1}, \mathsf{e1}'''\}, \mathsf{e1}') \parallel \mathsf{taskSnd}(\mathsf{e1}', \mathsf{c}_1.\mathsf{c} \neq \mathsf{null}, \mathsf{c}_1.\mathsf{c} := \mathsf{c}_1.\mathsf{c} + 1, \mathsf{m} : \tilde{\mathsf{exp}}_1, \mathsf{e1}'') \\ & \parallel \mathsf{xorSplit}(\mathsf{e1}'', \{(\mathsf{e1}''', \mathsf{c}_1.\mathsf{c} \leqslant \mathsf{n}), (\mathsf{e2}, \mathsf{default})\}) \parallel P' \end{aligned}$$

One-From-Many Receive Pattern

Informal Description: A party receives several messages.



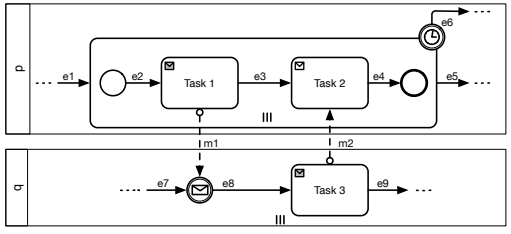
Textual Specification:

$$C = \text{miPool}(\text{p}, P) \parallel \text{pool}(\text{q}, Q) \parallel \text{pool}(\text{t}, T), \text{ where}$$

$$\begin{aligned}
Q \quad = \quad & \text{taskSnd}(\text{e1}, \text{exp}_1, \epsilon, \text{m}_{\textit{startTimer}} : \tilde{\text{exp}}_2, \text{e}') \parallel \text{xorJoin}(\{\text{e}', \text{e}^{\text{v}}\}, \text{e}'') \parallel \\
& \text{eventBased}(\text{e}'', (\text{m} : \tilde{\text{t}}_1, \text{e}'''), (\text{m}_{\textit{timeout}} : \tilde{\text{t}}_2, \text{e3})) \parallel \\
& \text{task}(\text{e}''', \text{c}_1.\text{c} \neq \text{null}, \text{c}_1.\text{c} := \text{c}_1.\text{c} + 1, \text{e}^{\text{iv}}) \parallel \\
& \text{xorSplit}(\text{e}^{\text{iv}}, \{(\text{e}^{\text{v}}, \text{c}_1.\text{c} \leqslant \text{n}), (\text{e2}, \text{default})\}) \parallel \text{xorJoin}(\{\text{e2}, \text{e3}\}, \text{e4}) \parallel Q' \\
T \quad = \quad & \text{startRcv}(\text{m}_{\textit{startTimer}} : \tilde{\text{t}}_3, \text{e5}) \parallel \text{taskSnd}(\text{e5}, \text{exp}_3, \epsilon, \text{m}_{\textit{timeout}} : \tilde{\text{exp}}_4, \text{e6}) \parallel \text{end}(\text{e6})
\end{aligned}$$

One-To-Many Send/Receive Pattern

Informal Description: A party sends requests to several other parties.



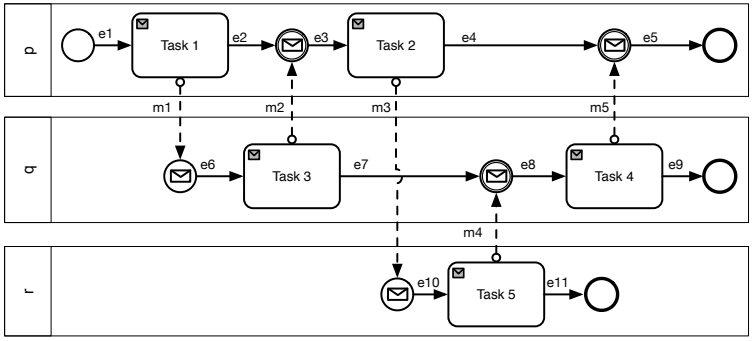
Textual Specification:

$$C = \text{pool}(\text{p}, P) \parallel \text{miPool}(\text{q}, Q), \text{ where}$$

$$\begin{aligned} Q &= \text{interRcv}(\text{e7}, \text{m1} : \tilde{\text{t}_1}, \text{e8}) \parallel \text{taskSnd}(\text{e8}, \text{exp}, A, \text{m2} : \tilde{\text{exp}}, \text{e9}) \parallel Q' \\ P &= \ldots \end{aligned}$$

Request With Referral Pattern

Informal Description: A party p makes a request to party q which delegate r to answer.



Textual Specification:

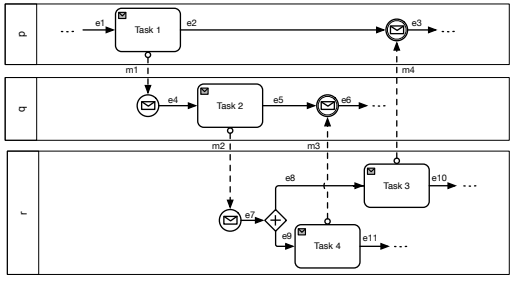
$$C = \text{pool}(\text{p}, P) \parallel \text{pool}(\text{q}, Q) \parallel \text{pool}(\text{r}, R), \text{ where}$$

$$\begin{aligned} P \quad = \quad & \text{start}(\text{e}_{enb}, \text{e1}) \parallel \text{taskSnd}(\text{e1}, \text{exp}_1, A_1, \text{m1} : \text{ex}\tilde{\text{p}}_2, \text{e2}) \parallel \text{interRcv}(\text{e2}, \text{m2} : \tilde{\text{t}_1}, \text{e3}) \parallel \\ & \text{taskSnd}(\text{e3}, \text{exp}_3, A_2, \text{m3} : \text{ex}\tilde{\text{p}}_4, \text{e4}) \parallel \text{interRcv}(\text{e4}, \text{m4} : \tilde{\text{t}_2}, \text{e5}) \parallel \text{end}(\text{e5}) \end{aligned}$$

Q and R are defined similarly.

Request With Referral Pattern

Informal Description: A party p makes a request to party q, which delegates the request processing to another party r.



Textual Specification:

$$C = \text{pool}(\text{p}, P) \parallel \text{pool}(\text{q}, Q) \parallel \text{pool}(\text{r}, R), \text{ where}$$

$$\begin{aligned} R \quad = \quad & \text{startRcv}(\text{m2}:\tilde{\text{t}_1}, \text{e7}) \parallel \text{andSplit}(\text{e7}, \{\text{e8}, \text{e9}\}) \parallel \\ & \text{taskSnd}(\text{e8}, \text{exp}_1, \epsilon, \text{m4}:\tilde{\text{exp}_2}, \text{e10}) \parallel \text{taskSnd}(\text{e9}, \text{exp}_3, \epsilon, \text{m3}:\tilde{\text{exp}_4}, \text{e11}) \parallel R' \end{aligned}$$

P and Q are defined similarly.

References

Flavio Corradini, Chiara Muzi, Barbara Re, Lorenzo Rossi, Francesco Tiezzi: Animating Multiple Instances in BPMN Collaborations: From Formal Semantics to Tool Support. BPM 2018: 83-101

Flavio Corradini, Chiara Muzi, Barbara Re, Lorenzo Rossi, Francesco Tiezzi: MIDA: Multiple Instances and Data Animator. BPM (Dissertation/Demos/Industry) 2018: 86-90

Chiara Muzi, Luise Pufahl, Lorenzo Rossi, Mathias Weske, Francesco Tiezzi: Formalising BPMN Service Interaction Patterns. PoEM 2018: 3-20