CoMPSeT – A Framework for Comparing Multiparty Session Types

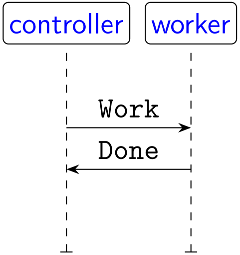
T. Ribeiro, J. Proença & M. Florido

EXPRESS/SOS @ CONFEST 2025

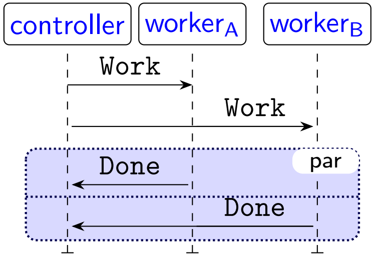
August 2025

Multiparty Session Types

It is all about ensuring communication *safety* and *liveness*



(Binary) Session Types



Multiparty Session Types

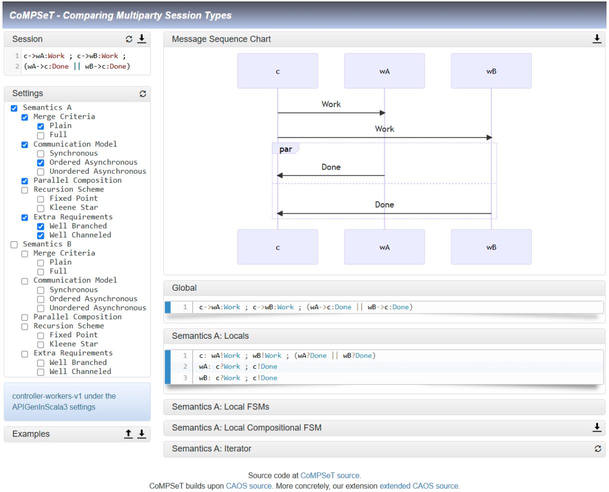
Semantical Differences

Table: Features mapping – ✓(present), ×(absent) or N/S (not specified)

Paper	**Merge criteria**	**Communication model**	**Parallel composition**	**Recursion scheme**	**Well-formedness requirements**
Yoshida & Gheri	plain & full	synchronous	×	fixed point	
Coppo et al.	plain	ordered asynchronous	×	fixed point	
Cledou et al.	plain	ordered asynchronous	✓	×	well-channelled
Jongmans & Proença	plain	ordered asynchronous	✓	**Kleene star** & fixed point	well-channelled
Guanciale & Tuosto	N/S	unordered asynchronous	N/S	N/S	N/S

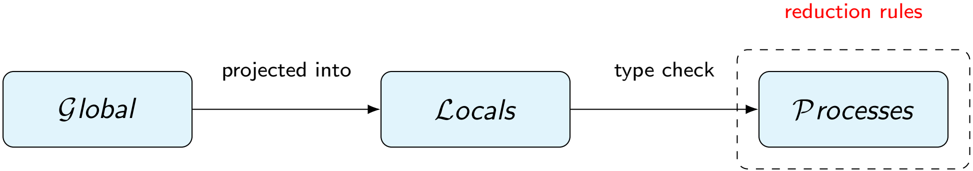
Subtle semantic differences are hard to understand and compare

What We Offer

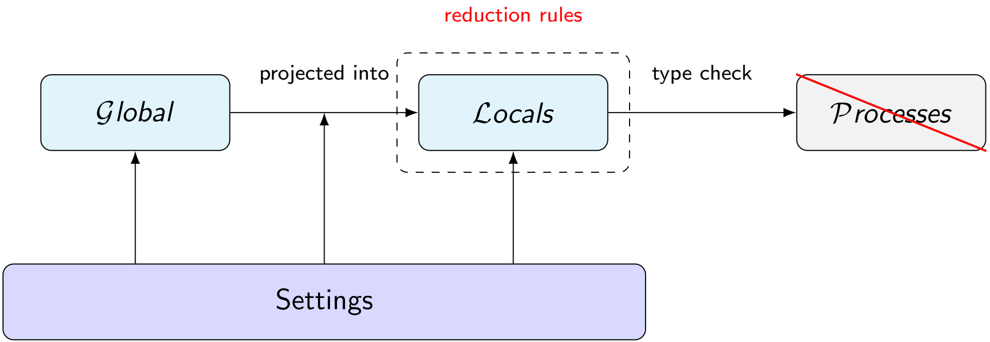


- CoMPSeT
  - open source & browser-executable
  - for comparing, visualising, and interacting with sessions and semantics
- Built on top of CAOS
  - *explained later*

Detailed Multiparty Session Types Framework



As Seen in CoMPSeT



Global and Local Types

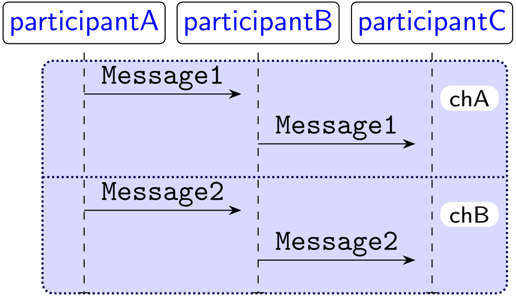
Global Type Grammar:

$$G ::= \; \mathsf{p} \rightarrow \mathsf{q} : \{\mathsf{t_i} \; ; \; G_i\}_{1 \leq i \leq n} \mid G_1 \; ; \; G_2 \mid G_1 \parallel G_2 \mid \mu X . G \mid X \mid (G)^* \mid \mathit{skip}$$

Local Type Grammar:

$$L ::= \mathsf{pq}!\{\mathsf{t_i} \; ; \; L_i\}_{1 \leq i \leq n} \mid \mathsf{pq}?\{\mathsf{t_i} \; ; \; L_i\}_{1 \leq i \leq n} \mid \ldots$$

Example With Branching



Global type:

$$\mathsf{pA} \to \mathsf{pB} : \{ \\ \quad \mathsf{m1} \,;\, \mathsf{pB} \to \mathsf{pC} : \mathsf{m1}, \\ \quad \mathsf{m2} \,;\, \mathsf{pB} \to \mathsf{pC} : \mathsf{m2} \\ \}$$

Local types:

$$L_{\mathsf{pA}} = \mathsf{pB}!\{\mathsf{m1},\ \mathsf{m2}\}$$

$$L_{\mathsf{pB}} = \mathsf{pA}?\{\mathsf{m1} \,;\, \mathsf{pC}!\mathsf{m1},\ \mathsf{m2} \,;\, \mathsf{pC}!\mathsf{m2}\}$$

$$L_{\mathsf{pC}} = \mathsf{pB}?\{\mathsf{m1},\ \mathsf{m2}\}$$

Variation Points in the Semantics

While projecting

$$p \to q : \{t_i \,;\, G_i\}_{1\le i\le n}\!\restriction_r \;=\; pq!\{t_i \,;\, (G_i\!\restriction_r)\}_{1\le i\le n} \qquad \text{if } p = r \neq q$$

$$p \to q : \{t_i \,;\, G_i\}_{1\le i\le n}\!\restriction_r \;=\; pq?\{t_i \,;\, (G_i\!\restriction_r)\}_{1\le i\le n} \qquad \text{if } p \neq r = q$$

$$p \to q : \{t_i \,;\, G_1\}_{1\le i\le n}\!\restriction_r \;=\; \text{merge}(\{G_i\!\restriction_r\}_{1\le i\le n}) \qquad \text{if } p \neq r \neq q$$

...

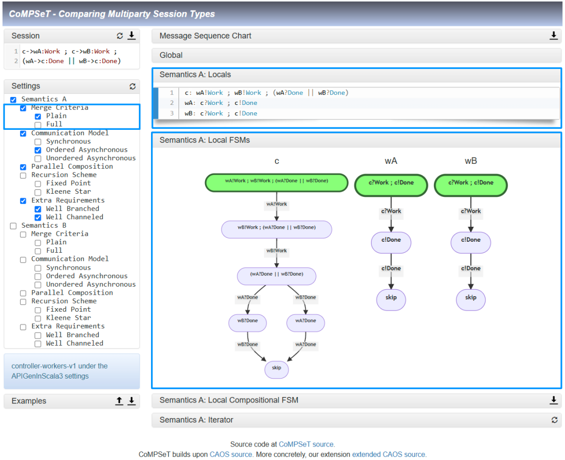
While running

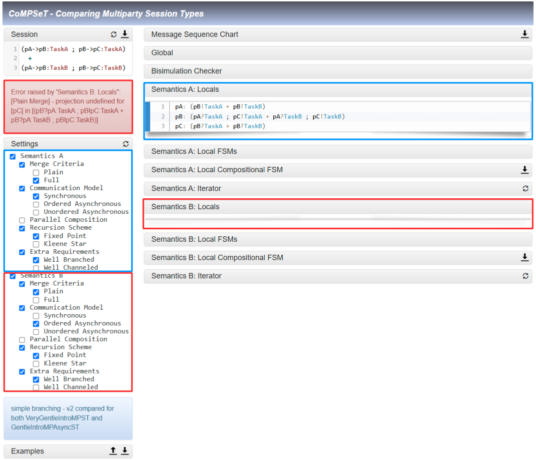
Assuming configurations such as $\langle M, p \rangle$, where M denotes $L_{p_1} \mid \ldots \mid L_{p_n}$ and $t_k \in \bigcup_{i=1}^{m} t_i$

$$\langle pq!\{t_i \,;\, L_i\}_{1\le i\le m} \mid M \,,\, p \cup \{pq \mapsto ts\}\rangle \xrightarrow{pq!t_k} \langle L_k \mid M \,,\, p \cup \{pq \mapsto ts\cdot t_k\}\rangle$$

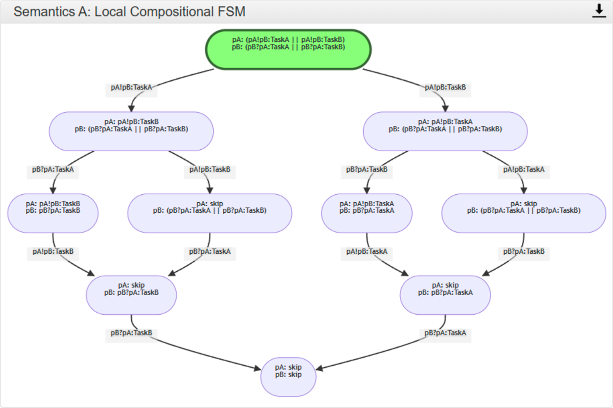
$$\langle pq?\{t_i \,;\, L_i\}_{1\le i\le m} \mid M \,,\, p \cup \{pq \mapsto t_k\cdot ts\}\rangle \xrightarrow{pq?t_k} \langle L_k \mid M \,,\, p \cup \{pq \mapsto ts\}\rangle$$

Merge Criteria

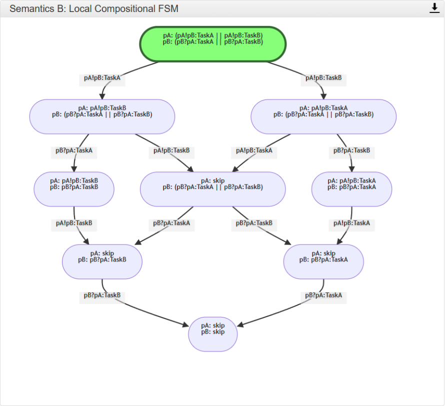




Communication Model



Ordered (Queue)



Unordered (Multiset)

For $\mathsf{pA} \to \mathsf{pB} : \mathsf{TaskA} \parallel \mathsf{pA} \to \mathsf{pB} : \mathsf{TaskB}$

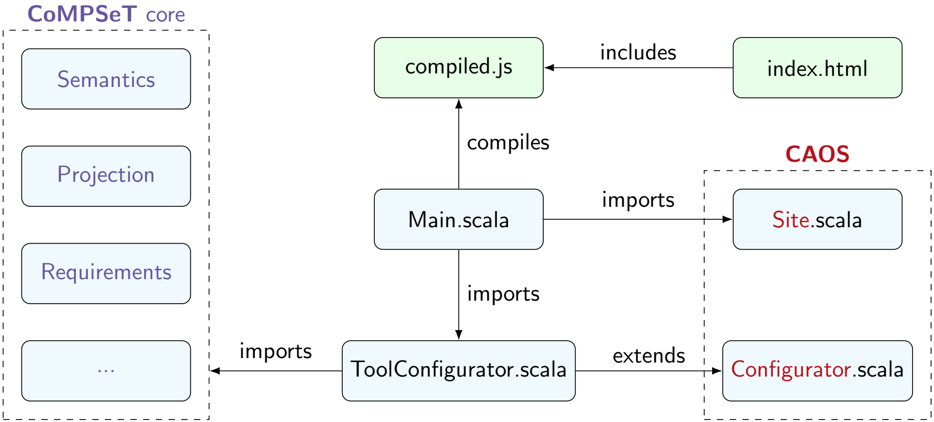
Covered Features

- Merge Criteria
- Communication Model
- Parallel Composition
- Recursion Scheme
- Extra Well-Formedness Requirements

Implementation Details

But Why CAOS?

It is all about *widgets* and ease of development



Wrap Up

What We Saw

- MPST are powerful yet fragmented
- Details may be hard to grasp
- CoMPSeT (and CAOS) enable comparisons over sessions and semantics

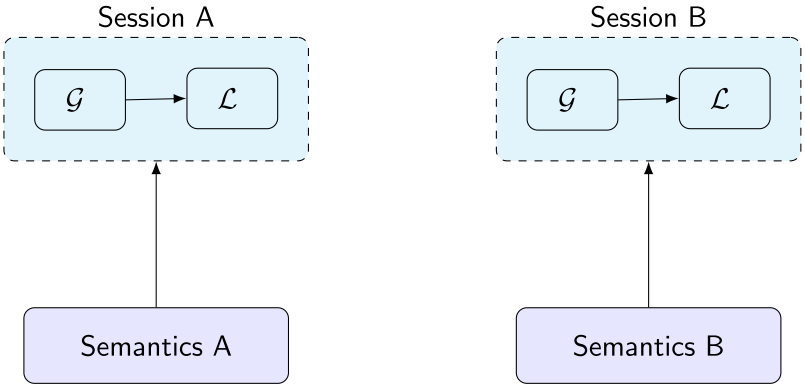
Future Work

- Additional feature assimilation
- API generation
- Formal proofs

CoMPSeT

Presented at EXPRESS/SOS @ CONFEST 2025
CoMPSeT: A Framework for Comparing Multiparty Session Types
T. Ribeiro, J. Proença & M. Florido

Comparing Sessions and Semantics



Projection

$$
\begin{array}{ll}
skip\lfloor_{\mathsf{r}} = skip & \\
X\lfloor_{\mathsf{r}} = X & \\
(\mu X.G)\lfloor_{\mathsf{r}} = \mu X.(G\lfloor_{\mathsf{r}}) & \text{if } \mathsf{r} \in participants\{G\} \\
(\mu X.G)\lfloor_{\mathsf{r}} = skip & \text{if } \mathsf{r} \notin participants\{G\} \\
(G)^*\lfloor_{\mathsf{r}} = (G\lfloor_{\mathsf{r}})^* & \text{if } \mathsf{r} \in participants\{G\} \\
(G)^*\lfloor_{\mathsf{r}} = skip & \text{if } \mathsf{r} \notin participants\{G\} \\
\mathsf{p} \to \mathsf{q} : \{\mathsf{t_i} \, ; G_i\}_{1\leq i\leq n}\lfloor_{\mathsf{r}} = \mathsf{pq}!\{\mathsf{t_i} \, ; (G_i\lfloor_{\mathsf{r}})\}_{1\leq i\leq n} & \text{if } \mathsf{p} = \mathsf{r} \neq \mathsf{q} \\
\mathsf{p} \to \mathsf{q} : \{\mathsf{t_i} \, ; G_i\}_{1\leq i\leq n}\lfloor_{\mathsf{r}} = \mathsf{pq}?\{\mathsf{t_i} \, ; (G_i\lfloor_{\mathsf{r}})\}_{1\leq i\leq n} & \text{if } \mathsf{p} \neq \mathsf{r} = \mathsf{q} \\
\mathsf{p} \to \mathsf{q} : \{\mathsf{t_i} \, ; G_1\}_{1\leq i\leq n}\lfloor_{\mathsf{r}} = merge(\{G_i\lfloor_{\mathsf{r}}\}_{1\leq i\leq n}) & \text{if } \mathsf{p} \neq \mathsf{r} \neq \mathsf{q} \\
(G_1 \, ; G_2)\lfloor_{\mathsf{r}} = (G_1\lfloor_{\mathsf{r}}) \, ; (G_2\lfloor_{\mathsf{r}}) & \\
(G_1 \parallel G_2)\lfloor_{\mathsf{r}} = (G_1\lfloor_{\mathsf{r}}) \parallel (G_2\lfloor_{\mathsf{r}}) & \\
\text{undefined} & \text{otherwise}
\end{array}
$$

Synchronous Semantics in CoMPSeT

$$\langle \text{pq}!\{\text{t}_\text{i} \,;\, L_{1_i}\}_{1\leq i\leq m_i} \mid \text{pq}?\{\text{t}_\text{j} \,;\, L_{2_j}\}_{1\leq j\leq m_j} \mid M\rangle \xrightarrow{\text{p}\to\text{q}:\text{t}_\text{k}} \langle L_{1_k} \mid L_{2_k} \mid M\rangle$$

Assumptions:

- The synchronous communication rule assumes no buffering mechanism, hence p is always empty and absent in the notation
- We assume that $\text{t}_\text{k} \in \bigcup_{i=1}^{m_i} \cap \bigcup_{j=1}^{m_j}$

Ordered Asynchronous Semantics in CoMPSeT

$$\langle \mathsf{pq}!\{\mathsf{t_i}\,;\,L_i\}_{1\leq i\leq m} \mid M\,,\,p \cup \{\mathsf{pq} \mapsto ts\}\rangle \xrightarrow{\mathsf{pq}!\mathsf{t_k}} \langle L_k \mid M\,,\,p \cup \{\mathsf{pq} \mapsto ts{\cdot}\mathsf{t_k}\}\rangle$$

$$\langle \mathsf{pq}?\{\mathsf{t_i}\,;\,L_i\}_{1\leq i\leq m} \mid M\,,\,p \cup \{\mathsf{pq} \mapsto \mathsf{t_k}{\cdot}ts\}\rangle \xrightarrow{\mathsf{pq}?\mathsf{t_k}} \langle L_k \mid M\,,\,p \cup \{\mathsf{pq} \mapsto ts\}\rangle$$

Assumptions:

- a buffer $p : (\mathbb{P} \times \mathbb{P}) \to \mathbb{T}^*$, mapping each pair sender-receiver to a sequence of data types in $\mathbb{T}$

Unordered Asynchronous Semantics in CoMPSeT

$$\langle \mathsf{pq}!\{\mathsf{t_i}\,;\,L_i\}_{1\leq i\leq m} \mid M\,,\,p\rangle \xrightarrow{\mathsf{pq}!\mathsf{t_k}} \langle L_k \mid M\,,\,p \cup (\mathsf{p},\mathsf{q},\mathsf{t_k})\rangle$$

$$\langle \mathsf{pq}?\{\mathsf{t_i}\,;\,L_i\}_{1\leq i\leq m} \mid M\,,\,p \cup (\mathsf{p},\mathsf{q},\mathsf{t_k})\rangle \xrightarrow{\mathsf{pq}?\mathsf{t_k}} \langle L_k \mid M\,,\,p\rangle$$

Assumptions:

- We assume $p \in \mathcal{M}(\mathbb{P} \times \mathbb{P} \times \mathbb{T})$, where $\mathcal{M}(X)$ denotes the set of all finite multisets over the set X.

Describing Widgets

```
lts(initialStateA, semanticsA, showStateA) /* for sync. */
lts(initialStateB, semanticsB, showStateB) /* for causal async. */
lts(initialStateC, semanticsC, showStateC) /* for non-causal async. /*
```

Native CAOS

```
enabledCommunicationModels(path).headOption.map {
  communicationModel =>
    val arguments = getArguments(root, communicationModel)
    lts(arguments.initialState, arguments.semantics, arguments.
        showState)
}
```

Extended CAOS (as used in CoMPSeT)

Allowing for

- Runtime widget variability
- Setting's concept and DSL
- Concise API f