Generalised Multiparty Session Types with Crash-Stop Failures

Adam D. Barwell[1] Alceste Scalas[2] Nobuko Yoshida[1] Fangyi Zhou[1]

[1]Imperial College London

[2]DTU Compute – Technical University of Denmark

Stardust Meeting
Kent

12th to 13th September 2022

Introduction

Well-typed processes enjoy *the Session Theorems*:

- ✓ Type Safety
- ✓ Protocol Conformance
- ✓ Deadlock-Freedom and Liveness

However …

- ✗ Most works assume a *perfect* world with no failures
- ✗ Failures occur in various ways
- ✗ Failures are difficult to model

In this work, we present a generalised session type theory with:

- » Crash-Stop Failures $\lightning$ and Detections $\odot$
- » Optional Reliability Assumptions $\mathcal{R}$
- » Type Level Model Checking $\Gamma \models \phi$
- » Guarantees from the Session Theorems $\xrightarrow{\checkmark}$

Processes

We use a session π-calculus[1]:

$$
\begin{array}{rcll}
c & ::= & x \mid s[\mathbf{p}] & \text{(variable or channel for session } s \text{ with role } \mathbf{p}\text{)} \\
P, Q & ::= & \mathbf{0} \mid (\nu s)\, P \mid P \mid Q & \text{(inaction, restriction, parallel composition)} \\
& \mid & c[\mathbf{q}] \oplus \mathtt{m}\langle d \rangle . P \quad \text{(where } \mathtt{m} \neq \textbf{crash}\text{)} & \text{(selection towards role } \mathbf{q}\text{)} \\
& \mid & c[\mathbf{q}] \& \{\mathtt{m}_i(x_i).P_i\}_{i \in I} & \text{(branching from role } \mathbf{q} \text{ with an index set } I \neq \emptyset\text{)}
\end{array}
$$

where

- v is a basic value (e.g. integers, strings, booleans)
- d is either a channel c or a basic value v
- $\mathtt{m}$ is a label, among which **crash** is a special label
- s is a session

[1]Some constructs are omitted for clarity of presentation, see full syntax in paper.

Crash-Stop Failures ↯

Intuition:

> *An active process may crash arbitrarily, and cease to interact with any other process afterwards.*

New process construct:

$$
\begin{array}{rcll}
P, Q & ::= & \cdots & \\
& | & s[\mathbf{p}]↯ & \text{(crashed channel endpoint)}
\end{array}
$$

Crash-Stop Failures $\lightning$

> *An active process may crash arbitrarily, and cease to interact with any other process afterwards.*

In operational semantics of processes:

$$[\text{R-}\lightning\oplus] \qquad P = s[\mathbf{p}][\mathbf{q}]\oplus \mathtt{m}\langle w\rangle.P' \;\longrightarrow\; \Pi_{j\in J} s_j[\mathbf{p}_j]\lightning \qquad \text{where } \{s_j[\mathbf{p}_j]\}_{j\in J} = \mathrm{fc}(P)$$

$$[\text{R-}\lightning\&] \qquad P = s[\mathbf{p}][\mathbf{q}]\&\{\mathtt{m}_i(x_i).P_i\}_{i\in I} \;\longrightarrow\; \Pi_{j\in J} s_j[\mathbf{p}_j]\lightning \qquad \text{where } \{s_j[\mathbf{p}_j]\}_{j\in J} = \mathrm{fc}(P)$$

where $\Pi_{i\in I} P_i$ is a shorthand notation of parallel compositions $P_1 \mid P_2 \mid \cdots \mid P_n$, and $\mathrm{fc}(P)$ is the set of free channel endpoints.

For example:

$$s[\mathbf{p}][\mathbf{q}]\oplus \mathtt{Foo}\langle s'[\mathbf{r}]\rangle.\mathbf{0} \rightarrow s[\mathbf{p}]\lightning \mid s'[\mathbf{r}]\lightning$$

Interacting with Crashed Endpoints

$s[\mathbf{p}][\mathbf{q}]\&\{\mathtt{m}_i(x_i).P_i\}_{i\in I}$

$s[\mathbf{q}]\lightning$

- Naively, we lose progress when a receiving process is waiting forever for a crashed endpoint
- We need additional rules for interacting with crashed endpoints, to complete our failure model

Crash Detection $\odot$

We use a special label **crash** to denote a *crash handling* branch, which is taken whenever a crash is detected:

$$[\text{R-}\odot] \qquad s[\mathbf{p}][\mathbf{q}]\&\{\mathtt{m}_i(x_i).P_i,\ \textbf{crash}.P'\}_{i\in I} \mid s[\mathbf{q}]\lightning \ \longrightarrow\ P' \mid s[\mathbf{q}]\lightning$$

Additionally, we need a rule to handle session endpoints sent to a crashed endpoint — the payload also becomes crashed:

$$[\text{R-}\lightning\,\mathtt{m}] \qquad s[\mathbf{p}]\lightning \mid s[\mathbf{q}][\mathbf{p}]\oplus\mathtt{m}\langle s'[\mathbf{r}]\rangle.Q' \ \longrightarrow\ s[\mathbf{p}]\lightning \mid s'[\mathbf{r}]\lightning \mid Q'$$

Session Types

We assign session types to channel endpoints:

$$
\begin{array}{llll}
B & ::= & \mathsf{int} \mid \mathsf{bool} \mid \mathsf{real} \mid \mathsf{unit} \mid \ldots & \text{(basic types)} \\
S & ::= & B \mid T & \text{(payload type: basic type or session type)} \\
T & ::= & \mathbf{p}\&\{\mathtt{m}_i(S_i).T_i\}_{i\in I} \mid \mathbf{p}\oplus\{\mathtt{m}_i(S_i).T_i\}_{i\in I} & \text{(external or internal choice, with } I \neq \emptyset\text{)} \\
 & \mid & \mu\mathbf{t}.T \mid \mathbf{t} \mid \mathsf{end} & \text{(recursion, type variable, or termination)} \\
U & ::= & T \mid \mathsf{stop} & \text{(session type or crash type)}
\end{array}
$$

in judgments such as:

$$\Gamma \vdash P$$

where

$$\Gamma \quad ::= \quad \emptyset \mid \Gamma, x{:}S \mid \Gamma, s[\mathbf{p}]{:}U$$

Typing Contexts Reductions in Multiparty Session Types

Typing contexts evolve as processes reduce.

For example:

$$\frac{\Gamma_1 \xrightarrow{s[\mathbf{p}]:\mathbf{q}\oplus \mathtt{m}(S)} \Gamma_1' \qquad \Gamma_2 \xrightarrow{s[\mathbf{q}]:\mathbf{p}\&\mathtt{m}(S')} \Gamma_2' \qquad S \leqslant S'}{\Gamma_1, \Gamma_2 \xrightarrow{s[\mathbf{p}][\mathbf{q}]\mathtt{m}} \Gamma_1', \Gamma_2'} \; [\Gamma\text{-}\oplus\&]$$

If $s[\mathbf{p}]$ in Γ_1 can send ($\oplus$) a message to $\mathbf{q}$, and $s[\mathbf{q}]$ in Γ_2 can receive ($\&$) that message from $\mathbf{p}$, with compatible types; then the combined context Γ_1, Γ_2 reduces with a label $s[\mathbf{p}][\mathbf{q}]\mathtt{m}$.

Typical Subject Reduction[2]:

Given $\Gamma \vdash P$ *with* $\mathrm{safe}(\Gamma)$, *and* $P \to P'$.
There exists Γ' *with* $\mathrm{safe}(\Gamma')$ *such that* $\Gamma' \vdash P'$ *and* $\Gamma \to^* \Gamma'$.

[2] Scalas and Yoshida. POPL '19. Less Is More: Multiparty Session Types Revisited

A Brief Example

$$s[\mathbf{p}] : \mathbf{q}\&\{\mathtt{data}.\mathbf{r}\oplus\mathtt{ok} \mid \mathbf{crash}.\mathbf{r}\oplus\mathtt{fail}\}$$

$$s[\mathbf{q}] : \mathbf{p}\oplus\mathtt{data} \qquad s[\mathbf{r}] : \mathbf{p}\&\{\mathtt{ok} \mid \mathtt{fail}\}$$

$$\downarrow s[\mathbf{q}][\mathbf{p}]\mathtt{data}$$

$$s[\mathbf{p}] : \mathbf{r}\oplus\mathtt{ok} \qquad s[\mathbf{q}] : \mathtt{end}$$

$$s[\mathbf{r}] : \mathbf{p}\&\{\mathtt{ok} \mid \mathtt{fail}\}$$

$$\downarrow s[\mathbf{p}][\mathbf{r}]\mathtt{ok}$$

$$s[\mathbf{p}] : \mathtt{end} \qquad s[\mathbf{q}] : \mathtt{end} \qquad s[\mathbf{r}] : \mathtt{end}$$

Modelling Crashes $\lightning$ and Detections $\odot$

$$\frac{T \nleqslant \mathsf{end}}{s[\mathbf{p}]{:}T \xrightarrow{s[\mathbf{p}]\lightning} s[\mathbf{p}]{:}\mathsf{stop}}\ [\Gamma\text{-}\lightning]$$

$$\frac{}{s[\mathbf{p}]{:}\mathsf{stop} \xrightarrow{s[\mathbf{p}]\mathsf{stop}} s[\mathbf{p}]{:}\mathsf{stop}}\ [\Gamma\text{-stop}]$$

$$\frac{\Gamma_1 \xrightarrow{s[\mathbf{q}]:\mathbf{p}\&\mathbf{crash}} \Gamma_1' \qquad \Gamma_2 \xrightarrow{s[\mathbf{p}]\mathsf{stop}} \Gamma_2'}{\Gamma_1, \Gamma_2 \xrightarrow{s[\mathbf{q}]\odot\mathbf{p}} \Gamma_1', \Gamma_2'}\ [\Gamma\text{-}\odot]$$

A Brief Example

$$s[\mathbf{p}] : \mathbf{q}\&\{\mathtt{data}.\mathbf{r}\oplus\mathtt{ok} \mid \textbf{crash}.\mathbf{r}\oplus\mathtt{fail}\}$$
$$s[\mathbf{q}] : \mathbf{p}\oplus\mathtt{data} \qquad s[\mathbf{r}] : \mathbf{p}\&\{\mathtt{ok} \mid \mathtt{fail}\}$$

$$\big\downarrow s[\mathbf{q}]\lightning$$

$$s[\mathbf{p}] : \mathbf{q}\&\{\mathtt{data}.\mathbf{r}\oplus\mathtt{ok} \mid \textbf{crash}.\mathbf{r}\oplus\mathtt{fail}\}$$
$$s[\mathbf{q}] : \mathsf{stop} \qquad s[\mathbf{r}] : \mathbf{p}\&\{\mathtt{ok} \mid \mathtt{fail}\}$$

$$\big\downarrow s[\mathbf{p}]\odot\mathbf{q}$$

$$s[\mathbf{p}] : \mathbf{r}\oplus\mathtt{fail} \qquad s[\mathbf{q}] : \mathsf{stop}$$
$$s[\mathbf{r}] : \mathbf{p}\&\{\mathtt{ok} \mid \mathtt{fail}\}$$

$$\big\downarrow s[\mathbf{p}][\mathbf{r}]\mathtt{fail}$$

$$s[\mathbf{p}] : \mathsf{end} \qquad s[\mathbf{q}] : \mathsf{stop} \qquad s[\mathbf{r}] : \mathsf{end}$$

Safety

safe is the *largest* predicate on typing contexts Γ such that, whenever safe(Γ):

If $s[\mathbf{p}]$ sends to $\mathbf{q}$, and $s[\mathbf{q}]$ receives from $\mathbf{p}$, then they shall communicate:

$$[\text{S-}\oplus\&] \qquad \Gamma \xrightarrow{s[\mathbf{p}]:\mathbf{q}\oplus \mathtt{m}(S)} \text{ and } \Gamma \xrightarrow{s[\mathbf{q}]:\mathbf{p}\& \mathtt{m}'(S')} \text{ implies } \Gamma \xrightarrow{s[\mathbf{p}][\mathbf{q}]\mathtt{m}}$$

If $s[\mathbf{p}]$ has stopped, and $s[\mathbf{q}]$ receives from $\mathbf{p}$, then the crash shall be detected:

$$[\text{S-}\lightning\&] \qquad \Gamma \xrightarrow{s[\mathbf{p}]\mathrm{stop}} \text{ and } \Gamma \xrightarrow{s[\mathbf{q}]:\mathbf{p}\& \mathtt{m}(S)} \text{ implies } \Gamma \xrightarrow{s[\mathbf{q}]\odot\mathbf{p}}$$

Safety holds for any context Γ' that Γ transitions into:

$$[\text{S-}\rightarrow_{\lightning}] \qquad \Gamma \rightarrow \Gamma' \text{ implies } \text{safe}(\Gamma')$$

Optional Reliability Assumptions $\mathcal{R}$

Surely, not everything can fail, right?

For each session s in a typing context Γ:
we can *optionally* assume a set of roles $\mathcal{R}$ to be *reliable*.

Consequences:

- Crash reductions of $s[\mathbf{r}]$ for a reliable role $\mathbf{r}$ are disregarded;
- Any role receiving from a reliable role $\mathbf{r}$ does not need a **crash** handling branch.

Revisiting the Session Theorems

With crash-stop failures and optional reliability assumptions, we need to revise our subject reduction theorem:

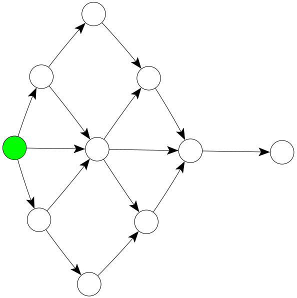
1. $\text{safe}(\Gamma)$ becomes $\text{safe}(\Gamma; s, \mathcal{R})$, where roles $\mathcal{R}$ in a session s are assumed reliable;
2. $\rightarrow$ becomes $\xrightarrow{\checkmark}$, where *assumption-abiding* reductions are considered.

Revised Subject Reduction:

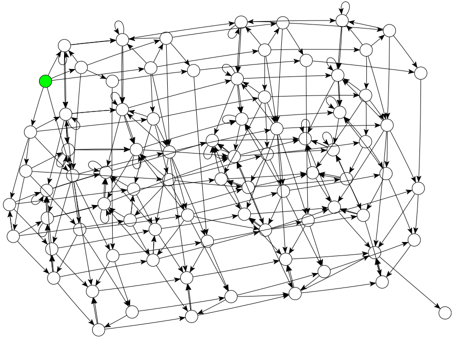
> *Given* $\Gamma \vdash P$ *with* $\forall s \in \Gamma : \exists \mathcal{R}_s : \text{safe}(\Gamma; s, \mathcal{R}_s)$, *and* $P \xrightarrow{\checkmark} P'$.
> *There exists* Γ' *with* $\forall s \in \Gamma' : \text{safe}(\Gamma'; s, \mathcal{R}_s)$ *such that* $\Gamma' \vdash P'$ *and* $\Gamma \rightarrow_{\lightning}^{*} \Gamma'$.

Other Session Theorems are revised in a similar way.

A Problem



becomes



Type Level Model Checking $\Gamma \models \phi$

Typing contexts Γ	become	models
Typing context properties $\varphi(\cdot)$	become	modal μ-calculus formulae ϕ

where $\varphi(\cdot)$ ranges over *safety*, *deadlock-freedom*, *terminating*, *never-terminating* and *liveness*.

We use the mCRL2 model checker, and our prototype is available on GitHub at `https://github.com/alcestes/mpstk-crash-stop`.

In the Paper

Pre-print available at:

`https://arxiv.org/abs/2207.02015`

We cover details of:

- type system: typing rules, and typing context transitions;
- how optional reliability is respected in considering process reductions;
- how properties are formulated as modal μ-calculus formulae;
- benchmarks that demonstrate viability of the model checking approach;
- ...

Ongoing Work: Integrate with Global Types

with Ping Hou and Nobuko Yoshida

B	$::=$	$\mathsf{int} \mid \mathsf{bool} \mid \mathsf{real} \mid \mathsf{unit} \mid \ldots$	Basic types
G	$::=$	$\mathbf{p}{\rightarrow}\mathbf{q}^{\dagger}: \{\mathtt{m_i}(B_i) \,.\, G_i\}_{i \in I}$	Transmission
	$\mid$	$\mathbf{p}^{\dagger}{\rightsquigarrow}\mathbf{q}{:}j\ \{\mathtt{m_i}(B_i) \,.\, G_i\}_{i \in I}$ (where $j \in I$)	Transmission en route (Runtime)
	$\mid$	$\mu\mathbf{t}.G \mid \mathbf{t} \mid \mathbf{end}$	Recursion, Type variable, Termination
$\dagger$	$::=$	$\cdot \mid \lightning$	Crash annotation (Runtime)

Crash annotations in global types mark crashed and live roles in a session.

Conclusion

We present a generalised session type theory with:

- Crash-Stop Failures $\lightning$ and Detections $\odot$
- Optional Reliability $\mathcal{R}$
- Type Level Model Checking $\Gamma \models \phi$
- Guarantees from the Session Theorems $\xrightarrow{\checkmark}$

Future work:

- Investigate Different Failure Models

See full version of the paper at `https://arxiv.org/abs/2207.02015`

See our prototype at `https://github.com/alcestes/mpstk-crash-stop`