Unreliability in Practical Subclasses of Communicating Systems (FSTTCS25)

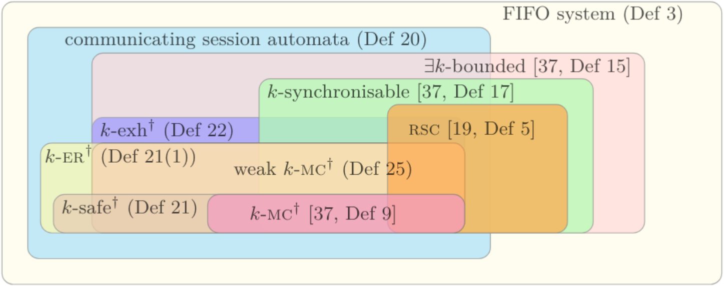
Amrita Suresh, **Nobuko Yoshida**, University of Oxford, UK
Presented by: Gabriele Genovese

13/11/2025

Introduction

- FIFO system $\longrightarrow$ Turing complete
- Perfect channels often assumed
- Aim: study unreliability...
  - interferences
  - crash of processes
- ...in two Practical Subclasses of FIFO (half-duplex) system:
  - Realisable with Synchronous Communication (RSC)
  - k-multiparty compatibility (k-MC)

Classes of communication systems



Aim

Do inclusion (or membership) of these subclasses remains decidable?

Do the complexity remains the same?

Can we translate the results in the MPST world?

Aim - Spoiler

Do inclusion (or membership) of these subclasses remains decidable? **Yes.**

Do the complexity remains the same? **Yes.**

Can we translate the results in the MPST world? **Yes.**

Contributions

- i-RSC and *weak* k-MC with *interferences* is decidable
- i-RSC and *weak* k-MC with *crash failures* is decidable
- *Translation* from local types (MPST) to crash-handling FIFO systems with proof of preserving trace semantics
- Evaluation of protocols with tools

Preliminaries

▶ **Definition 2** (FIFO automaton). *A FIFO automaton $\mathcal{A}_{\mathsf{p}}$, associated with p, is defined as $\mathcal{A}_{\mathsf{p}} = (Q_{\mathsf{p}}, \delta_{\mathsf{p}}, q_{0\mathsf{p}})$ where: Q_{p} is the finite set of control-states, $\delta_{\mathsf{p}} \subseteq Q_{\mathsf{p}} \times \mathsf{Act} \times Q_{\mathsf{p}}$ is the transition relation, and $q_{0\mathsf{p}} \in Q_{\mathsf{p}}$ is the initial control-state.*

▶ **Definition 3** (FIFO system). *A FIFO system $\mathcal{S} = (\mathcal{A}_{\mathsf{p}})_{\mathsf{p} \in \mathbb{P}}$ is a set of communicating FIFO automata. A configuration of $\mathcal{S}$ is a pair $\gamma = (\vec{q}\,; \vec{w})$ where $\vec{q} = (q_{\mathsf{p}})_{\mathsf{p} \in \mathbb{P}}$ is called the* global state *with $q_{\mathsf{p}} \in Q_{\mathsf{p}}$ being one of the local control-states of $\mathcal{A}_{\mathsf{p}}$, and where $\vec{w} = (w_{\mathsf{pq}})_{\mathsf{pq} \in Ch}$ with $w_{\mathsf{pq}} \in \Sigma^*$.*

Interferences ($\succeq$)

Reflexivity	*Transitivity*	*Additivity*	*Integrity*	*Non-expansion*
$\dfrac{a \in \Sigma}{a \succeq a}$	$\dfrac{w \succeq w' \quad w' \succeq w''}{w \succeq w''}$	$\dfrac{w_1 \succeq w_1' \quad w_2 \succeq w_2'}{w_1 \cdot w_2 \succeq w_1' \cdot w_2'}$	$\dfrac{\varepsilon \succeq w}{w = \varepsilon}$	$\dfrac{w \succeq w'}{\lvert w \rvert \geq \lvert w' \rvert}$

Interferences ($\succeq$)

$$\frac{a \in \Sigma}{a \succeq a}\;\textit{Reflexivity} \qquad \frac{w \succeq w' \quad w' \succeq w''}{w \succeq w''}\;\textit{Transitivity} \qquad \frac{w_1 \succeq w_1' \quad w_2 \succeq w_2'}{w_1 \cdot w_2 \succeq w_1' \cdot w_2'}\;\textit{Additivity} \qquad \frac{\varepsilon \succeq w}{w = \varepsilon}\;\textit{Integrity} \qquad \frac{w \succeq w'}{|w| \geq |w'|}\;\textit{Non-expansion}$$

- Additivity: failures can happen at any part of the words

Interferences ($\succeq$)

$$\frac{a \in \Sigma}{a \succeq a}\,\textit{Reflexivity} \qquad \frac{w \succeq w' \;\; w' \succeq w''}{w \succeq w''}\,\textit{Transitivity} \qquad \frac{w_1 \succeq w_1' \;\; w_2 \succeq w_2'}{w_1 \cdot w_2 \succeq w_1' \cdot w_2'}\,\textit{Additivity} \qquad \frac{\varepsilon \succeq w}{w = \varepsilon}\,\textit{Integrity} \qquad \frac{w \succeq w'}{|w| \geq |w'|}\,\textit{Non-expansion}$$

- Additivity: failures can happen at any part of the words
- Integrity: ϵ is the least word

Interferences ($\succeq$)

$$\frac{a \in \Sigma}{a \succeq a}\;\textit{Reflexivity} \qquad \frac{w \succeq w' \quad w' \succeq w''}{w \succeq w''}\;\textit{Transitivity} \qquad \frac{w_1 \succeq w_1' \quad w_2 \succeq w_2'}{w_1 \cdot w_2 \succeq w_1' \cdot w_2'}\;\textit{Additivity} \qquad \frac{\varepsilon \succeq w}{w = \varepsilon}\;\textit{Integrity} \qquad \frac{w \succeq w'}{|w| \geq |w'|}\;\textit{Non-expansion}$$

- Additivity: failures can happen at any part of the words
- Integrity: ϵ is the least word
- Non-expansion: $\succeq$ preserves the size of words

Type of interferences

Type of interferences

- *Lossiness*: message is lost during transmission ($a \succeq \epsilon$)

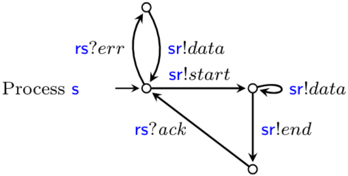
Type of interferences

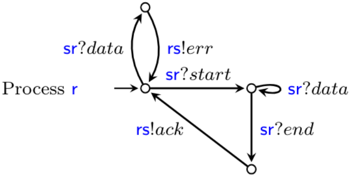
- *Lossiness*: message is lost during transmission ($a \succeq \epsilon$)
- *Corruption*: message is transformed during transmission ($a \succeq b$)

Type of interferences

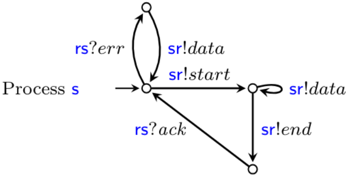
- *Lossiness*: message is lost during transmission ($a \succeq \epsilon$)
- *Corruption*: message is transformed during transmission ($a \succeq b$)
- *Out-of-order*: messages arrives at different time ($a \cdot b \succeq b \cdot a$)

Example


Process s
rs?err
sr!data
sr!start
sr!data
rs?ack
sr!end

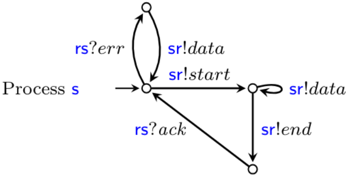

Process r
sr?data
rs!err
sr?start
sr?data
rs!ack
sr?end

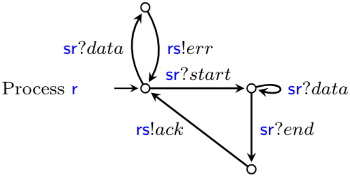
Example



- Corruption: $sr!start.sr?start.sr!data.sr?end.rs!ack.sr!data.$

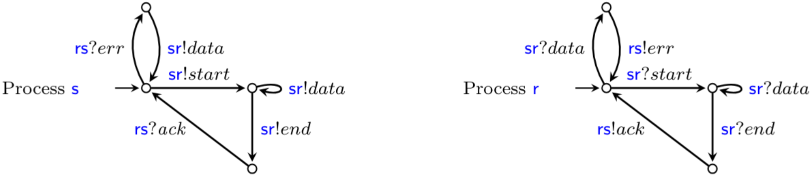
Example





- Corruption: *sr!start.sr?start.sr!data.sr?end.rs!ack.sr!data.*
- Lossiness: *sr!start.sr?start.sr!data.sr?data.sr!end*

Example



- Corruption: $sr!start.sr?start.sr!data.sr?end.rs!ack.sr!data$.
- Lossiness: $sr!start.sr?start.sr!data.sr?data.sr!end$
- Out-of-order: $sr!start.sr?start.sr!data.sr!end.sr?end.rs!ack.$ $rs?ack.sr?data.rs!err.rs?err$

i-RSC

- Extend the definition of *matching pairs* to include interference:
  - the message can be different (corruption)
  - a "receive" action is not strictly after a "send" action (out-of order)
- An *interaction* is either a matching pair, or a singleton (lossiness)

i-RSC

- **Definition 7** (Matching pair with interference). *Given an execution $e = a_1 \dots a_n$, if there exists a channel* pq*, messages $m, m' \in \Sigma$ and $j, j', k, k' \in \{1, \dots, n\}$ where $j < j'$, and the following four conditions:*

  (1) $a_j = \mathsf{pq}!m$; (2) $a_{j'} = \mathsf{pq}?m'$; (3) a_j *is the k-th send action to* pq *in e; and* (4) $a_{j'}$ *is the k'-th receive action on* pq *in e, then we say that $\{j, j'\} \subseteq \{1, \dots, n\}$ is a* matching pair with interference, *or i-matching pair.*

- **Definition 9** (Interaction). *An* interaction *of e is either a (perfect or i-) matching pair, or a singleton $\{j\}$ such that a_j is a send action and j does not belong to any matching pair (such an interaction is called unmatched send).*

Example

$$e = a_1 \ldots a_5 = pq!a \cdot qp!b \cdot qp?b \cdot pq!c \cdot pq?c$$

Two *valid* communication:

$\mathrm{Comm}(e) = \{\nu_1, \nu_2\}$

Example

$$e = a_1 \ldots a_5 = pq!a \cdot qp!b \cdot qp?b \cdot pq!c \cdot pq?c$$

Two *valid* communication:

- $\nu_1 = \{\{1,5\},\{2,3\},\{4\}\}$

$\mathrm{Comm}(e) = \{\nu_1, \nu_2\}$

Example

$$e = a_1 \dots a_5 = pq!a \cdot qp!b \cdot qp?b \cdot pq!c \cdot pq?c$$

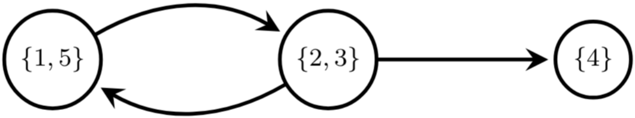
Two *valid* communication:

- $\nu_1 = \{\{1,5\},\{2,3\},\{4\}\}$
- $\nu_2 = \{\{1\},\{2,3\},\{4,5\}\}$

$\mathrm{Comm}(e) = \{\nu_1, \nu_2\}$

Conflict graph and borderline violation

- Characterise causally equivalent executions, using conflict graph



- *Borderline violation*: "minimal counter-example" for non-RSC behaviour (regular language)

Conflict graph and borderline violation

Definition 11 (Conflict graph). *Given an execution* (e, ν)*, the conflict graph* cgraph(e, ν) *the directed graph* $(\nu, \rightarrow_{e,\nu})$ *where for all interactions* $\chi_1, \chi_2 \in \nu$, $\chi_1 \rightarrow_{e,\nu} \chi_2$ *if there is* $\in \chi_1$ *and* $j_2 \in \chi_2$ *such that* $j_1 \prec_{e,\nu} j_2$.

▶ **Definition 15** (Borderline violation). *An execution* (e, ν) *is a borderline violation if* (1) (e, ν) *is not causally equivalent to an i-*RSC *execution,* (2) $e = e' \cdot \mathsf{c}?m$ *for some execution* e' *such that* (a) *for all* $\nu' \in \mathsf{Comm}(e')$, (e', ν') *is equivalent to an i-*RSC *execution and* (b) *there exists* $\nu_1 \in \mathsf{Comm}(e')$ *such that* (e', ν_1) *is an i-*RSC *execution.*

i-RSC

All matching pairs in valid communication are of the form $\{j, j+1\}$.

▶ **Definition 12** (i-RSC system). *An execution (e, ν) is i-RSC if all matching pairs in ν are of the form $\{j, j+1\}$. A system $\mathcal{S}$ is i-RSC if for all tuples (e, ν) such that $e \in \mathsf{executions}(\mathcal{S})$ and $\nu \in \mathit{Comm}(e)$, we have $\mathsf{cgraph}(e, \nu) = \mathsf{cgraph}(e', \nu')$ where (e', ν') is an i-RSC execution.*

ν_2 is an i-RSC execution.

▶ **Theorem 19.** *Given a system $\mathcal{S}$ of size n, deciding whether it is an i-RSC system can be done in time $\mathcal{O}(n^{|\mathbb{P}|+2}|Ch|^5 \times 2^{|Ch|} \times |\Sigma|^2)$.*

Lemmas

An execution is an *i*-RSC execution iff the conflict graph is acyclic:

▶ **Lemma 14.** *An execution (e, ν) is causally equivalent to an i-RSC execution iff the associated conflict graph* $\mathsf{cgraph}(e, \nu)$ *is acyclic.*

A system is *i*-RSC iff every execution is not a borderline violation

▶ **Lemma 16.** *$\mathcal{S}$ is i-RSC if and only if for all $e \in \mathsf{executions}(\mathcal{S})$ and $\nu \in \mathit{Comm}(e)$, (e, ν) is not a borderline violation.*

The language of borderline violation is regular:

▶ **Lemma 17.** *Let $\mathcal{S}$ with $\mathsf{product}(\mathcal{S}) = (Q, \Sigma, Ch, \mathsf{Act}, \delta, q_o)$. There is a non-deterministic finite state automaton $\mathcal{A}_{bv}$ computable in time $\mathcal{O}(|Ch|^3|\Sigma|^2)$ such that $\mathcal{L}(\mathcal{A}_{bv}) = \{e \in \mathsf{Act}_{nr}^*.\mathsf{Act}_? \mid \exists \nu \in \mathit{Comm}(e)$ such that (e, ν) is a borderline violation$\}$.*

The subset of executions that begin with an *i*-RSC prefix and terminate with a reception is regular:

▶ **Lemma 18.** *Let $\mathcal{S}$ be a FIFO system. There exists a non-deterministic finite state automaton $\mathcal{A}_{rsc}$ over $\mathsf{Act}_{nr} \cup \mathsf{Act}_?$ such that $\mathcal{L}(\mathcal{A}_{rsc}) = \{e \cdot \mathsf{pq}?m \in \mathsf{Act}_{nr}^*.\mathsf{Act}_? \mid e \cdot \mathsf{pq}?m \in \mathsf{executions}(\mathcal{S})$ and $\exists \nu \in \mathit{Comm}(e)$ such that (e, ν) is an i-RSC execution$\}$, which can be constructed in time $\mathcal{O}(n^{|\mathbb{P}|+2}|Ch|^2 \times 2^{|Ch|})$, where n is the size of $\mathcal{S}$.*

k-Multiparty Compatibility

Definition of k-MC with two properties:

k-Multiparty Compatibility

Definition of k-MC with two properties:

- *k-safety*:

k-Multiparty Compatibility

Definition of k-MC with two properties:

- *k-safety*:
  - k-ER: **eventual reception**

k-Multiparty Compatibility

Definition of k-MC with two properties:

- *k-safety*:
  - k-ER: **eventual reception**
  - k-PG: **progress**

k-Multiparty Compatibility

Definition of k-MC with two properties:

- *k-safety*:
  - k-ER: **eventual reception**
  - k-PG: **progress**
- *k-exhaustivity*: all k-reachable configurations, whenever a send action is enabled, it can be fired within a k-bounded execution

Weak k-MC with interferences is decidable

- k-safety is a too **strong** condition: *remove* progress because in case of *lossiness* it cannot be guaranteed.

Weak k-MC with interferences is decidable

- k-safety is a too **strong** condition: *remove* progress because in case of *lossiness* it cannot be guaranteed.

- Definition of **weak** k-MC: A communicating system is weakly k-mc, if it satisfies k-ER and is k-exhaustive.

Weak k-MC with interferences is decidable

- k-safety is a too **strong** condition: *remove* progress because in case of *lossiness* it cannot be guaranteed.
- Definition of **weak** k-MC: A communicating system is weakly k-mc, if it satisfies k-ER and is k-exhaustive.
- Thm: given a system with interference, checking the weak k-mc property is decidable and PSPACE-complete.

FIFO system with crash

- Declare a set of *reliable* processes $\mathcal{R}$ (can be empty).

FIFO system with crash

- Declare a set of *reliable* processes $\mathcal{R}$ (can be empty).
- Not reliable processes can crash

FIFO system with crash

- Declare a set of *reliable* processes $\mathcal{R}$ (can be empty).
- Not reliable processes can crash
  - send a **broadcast** when crash

FIFO system with crash

- Declare a set of *reliable* processes $\mathcal{R}$ (can be empty).
- Not reliable processes can crash
  - send a **broadcast** when crash
- A process has crash-handling behaviour if it has:

FIFO system with crash

- Declare a set of *reliable* processes $\mathcal{R}$ (can be empty).
- Not reliable processes can crash
  - send a **broadcast** when crash
- A process has crash-handling behaviour if it has:
  - Crash handling (CH): no deadlocked processes

FIFO system with crash

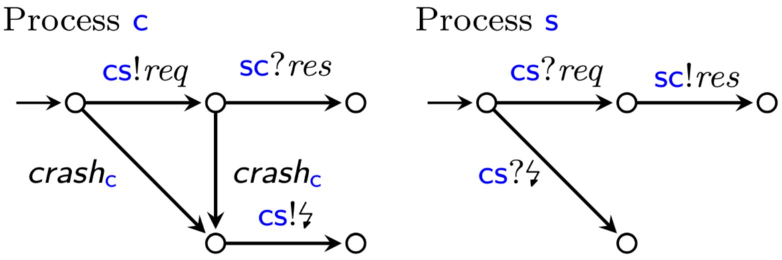
- Declare a set of *reliable* processes $\mathcal{R}$ (can be empty).
- Not reliable processes can crash
  - send a **broadcast** when crash
- A process has crash-handling behaviour if it has:
  - Crash handling (CH): no deadlocked processes
  - Crash broadcast (CB): if crash, send broadcast

FIFO system with crash

- Declare a set of *reliable* processes $\mathcal{R}$ (can be empty).
- Not reliable processes can crash
  - send a **broadcast** when crash
- A process has crash-handling behaviour if it has:
  - Crash handling (CH): no deadlocked processes
  - Crash broadcast (CB): if crash, send broadcast
  - Crash redundancy (CR): empty channels

Example of crash handling behaviour

Server is reliable, *client* is not.



Results for RSC and weak k-MC

- Lem: checking crash-handling is decidable.

Results for RSC and weak k-MC

- Lem: checking crash-handling is decidable.
- Lem: boundedness problem remains undecidable in crash-handling systems.

Results for RSC and weak k-MC

- Lem: checking crash-handling is decidable.
- Lem: boundedness problem remains undecidable in crash-handling systems.
- **Thm:** checking inclusion to the RSC class is decidable in crash-handling systems.
  - Proof adapted from "Cinzia Di Giusto, Loïc Germerie Guizouarn, and Etienne Lozes. Multiparty half-duplex systems and synchronous communications".

Results for RSC and weak k-MC

- Lem: checking crash-handling is decidable.
- Lem: boundedness problem remains undecidable in crash-handling systems.
- **Thm:** checking inclusion to the RSC class is decidable in crash-handling systems.
  - Proof adapted from "Cinzia Di Giusto, Loïc Germerie Guizouarn, and Etienne Lozes. Multiparty half-duplex systems and synchronous communications".
- **Thm:** checking weak k-MC is PSPACE for crash-handling system generated from communicating session automata.

Local Types with crash-handling

- **stop** type to denote crashed processes
- crash-handling branch (**catch**) in an external choice branch

$$
\begin{array}{llll}
S, T & ::= & \mathsf{p}?\{m_i.T_i\}_{i\in I} \mid \mathsf{p}!\{m_i.T_i\}_{i\in I} & \text{(external choice, internal choice)} \\
 & \mid & \mu \mathbf{t}.T \mid \mathbf{t} \mid \mathsf{end} \mid \mathsf{stop} & \text{(recursion, type variable, end, crash)}
\end{array}
$$

LTS over Local Type

▶ **Definition 34** (LTS over local types). *The relation $T \xrightarrow{a} T'$ for the local type of role* p *is defined as:*

$$
\begin{array}{llcll}
\text{[LR1]} & \mathsf{q}\dagger\{m_i.T_i\}_{i\in I} & \xrightarrow{\mathsf{pq}\dagger m_k} & T_k, & \textit{where } \dagger \in \{!,?\} \textit{ and } m_k \neq \textit{catch}. \\
\text{[LR2]} & T[\mu\mathbf{t}.T/\mathbf{t}] \xrightarrow{a} T' & \Longrightarrow & \mu\mathbf{t}.T \xrightarrow{a} T' & \\
\text{[LR3]} & \mathsf{q}\dagger\{m_i.T_i\}_{i\in I} & \xrightarrow{\textit{crash-broadcast}_{\mathsf{p}}(\lightning)} & \mathsf{stop}, & \textit{where } \dagger \in \{!,?\}. \\
\text{[LR4]} & \mathsf{q}?\{m_i.T_i\}_{i\in I} & \xrightarrow{\mathsf{qp}?\lightning} & T_k, & \textit{if } m_k = \textit{catch}. \\
\text{[LR5]} & T & \xrightarrow{\mathsf{qp}?\lightning} & T, & \forall \mathsf{q} \in \mathbb{P} \setminus \{\mathsf{p}\} \textit{ for } T \in \{\mathsf{stop}, \mathsf{end}\}.
\end{array}
$$

LTS over Local Type

LR1 and [LR2] are standard output/input and recursion rules,

LR3 aid the crash of a process (CB),

LR4 main rule to enter crash-handling branch (CH),

LR5 read all dangling crash messages (CR).

Translation from Local Types to FIFO automata

▶ **Definition 37** (From local types to FIFO automata). *Let T_0 be the local type of participant* p. *The automaton corresponding to T_0 is $\mathcal{A}(T_0) = (Q, \delta, q_0)$ where:*

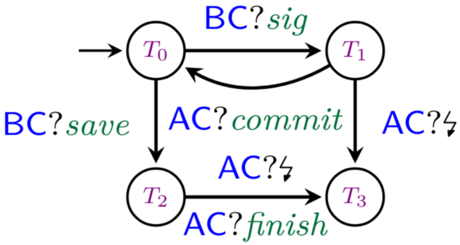
1. $Q = \{T' \mid T' \in T_0, T' \neq \mathbf{t}, T' \neq \mu\mathbf{t}.T\} \cup \{q_{crash}\} \cup \{q_{send,\mathsf{r}} \mid \mathsf{r} \in \mathbb{P} \setminus \{\mathsf{p}\}\}$
2. $q_0 = \mathrm{strip}(T_0)$;
3. δ *is the smallest set of transitions such that* $\forall T \in Q$:
   a. *If* $T = \mathsf{q}\dagger\{m_i.T_i\}_{i\in I}$ *and* $k \in I$, $m_k \neq \mathit{catch}$, *and* $\dagger \in \{!, ?\}$
   $$\begin{cases} (T, \mathsf{pq} \dagger m_k, \mathrm{strip}(T_k)) \in \delta & \textit{if } T_k \neq \mathbf{t} \\ (T, \mathsf{pq} \dagger m_k, \mathrm{strip}(T')) \in \delta & \textit{if } T_k = \mathbf{t} \textit{ with } \mu\mathbf{t}.T' \in T_0. \end{cases}$$
   b. *If* $T = \mathsf{q}?\{m_i.T_i\}_{i\in I}$ *with* $k \in I$, $m_k = \mathit{catch}$
   $$\begin{cases} (T, \mathsf{qp}?\lightning, \mathrm{strip}(T_k)) \in \delta & \textit{if } T_k \neq \mathbf{t} \\ (T, \mathsf{qp}?\lightning, \mathrm{strip}(T')) \in \delta & \textit{if } T_k = \mathbf{t} \textit{ with } \mu\mathbf{t}.T' \in T_0. \end{cases}$$
   c. *If* $T \notin \{\mathsf{stop}, \mathsf{end}\}$, *then* $(T, \textit{crash-broadcast}_{\mathsf{p}}(\lightning), \mathsf{stop}) \subseteq \delta$ *where*
      i. $(T, \textit{crash}, q_{crash}) \in \delta$
      ii. $(q_{crash}, \mathsf{pr}_1!\lightning, q_{send,\mathsf{r}_1}) \in \delta$
      iii. $(q_{send,\mathsf{r}_i}, \mathsf{pr}_{i+1}!\lightning, q_{send,\mathsf{r}_{i+1}}) \in \delta \ \forall i \in \{1, \ldots, n-2\}$, *where* $n = |Ch_{o,\mathsf{p}}|$
      iv. $(q_{send,\mathsf{r}_{n-1}}, \textit{crash}, \mathsf{stop}) \in \delta$
   d. *If* $T \in \{\mathsf{stop}, \mathsf{end}\}$, *then* $(T, \mathsf{qp}?\lightning, T) \in \delta$ *for all* $\mathsf{q} \in \mathbb{P} \setminus \{\mathsf{p}\}$.

where $\mathrm{strip}(T) \stackrel{\text{def}}{=} \mathrm{strip}(T')$ *if* $T = \mu\mathbf{t}.T'$; *otherwise* $\mathrm{strip}(T) \stackrel{\text{def}}{=} T$.

Example of Local Type with crash-handling

▶ **Example 36.** Let $\mathbb{P} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$ and $\mathcal{R} = \{\mathsf{B}, \mathsf{C}\}$. Consider a local type of C: $T = \mu \mathbf{t}.\mathsf{B}?\{sig.\mathsf{A}?\{commit.\mathbf{t},\ catch.\mathsf{end}\},\ save.\mathsf{A}?\{finish.\mathsf{end},\ catch.\mathsf{end}\}\}$.

Then, the set of all $T' \in T$ is $\{T, \mathsf{B}?\{sig.\mathsf{A}?\{commit.\mathbf{t},\ catch.\mathsf{end}\}\}, \mathsf{A}?\{commit.\mathbf{t}\}, \mathsf{B}?\{save.\mathsf{A}?\{finish.\mathsf{end},\ catch.\mathsf{end}\}\}, \mathsf{A}?\{catch.\mathsf{end}\}, \mathsf{A}?\{finish.\mathsf{end}\}, \mathsf{end}, \mathbf{t}\}$.



Results for Local Types

▶ **Lemma 39.** *Assume* T_{p} *is a local type. Then* $\mathcal{A}(T_{\mathsf{p}})$ *is deterministic, directed and has no mixed states. Moreover,* $T_{\mathsf{p}} \approx \mathcal{A}(T_{\mathsf{p}})$*, i.e.* $\forall \phi, \phi \in \mathsf{executions}(T_{\mathsf{p}}) \Leftrightarrow \phi \in \mathsf{executions}(\mathcal{A}(T_{\mathsf{p}}))$.

▶ **Theorem 40.** *The FIFO system generated from the translation of crash-stop session types is a crash-handling system. Moreover, it is decidable to check inclusion to the* RSC *and* k-WMC *classes.*

Experimental evaluation

Tools used and characteristics:

- RSC-checker: ReSCu
- k-mc-checker: kmc (added out-of-order)
- lossiness modelled with self-loops in automata
- corruption modelled with sending of arbitrary messages
- examples taken from referenced paper
- added a test for the Paxos algorithm

Table

Protocol	No errors		Out of order		Lossiness				Corruption			
	k-MC	RSC	k-MC	RSC	k-exh	k-ER	k-PG	RSC	k-exh	k-ER	k-PG	RSC
Alternating Bit [43]	yes	yes	yes	yes	yes	yes	no	yes	yes	yes	no	yes
Alternating Bit [7]	yes	no	yes	yes	yes	yes	no	yes	yes	yes	no	yes
Bargain [36]	yes	yes	yes	yes	yes	yes	no	yes	yes	no	no	yes
Client-Server-Logger [37]	yes	no	yes	no	yes	yes	no	yes	yes	yes	no	yes
Cloud System v4 [27]	yes	yes	yes	no	no	no	no	yes	no	no	no	no
Commit protocol [11]	yes	yes	yes	yes	yes	no	no	yes	yes	no	no	yes
Dev System [42]	yes	yes	yes	yes	yes	no	no	yes	yes	no	no	yes
Elevator [11]	yes	no	yes	no	yes	yes	no	no	no	TO	no	no
Elevator-dashed [11]	yes	no	yes	no	no	no	no	no	no	TO	no	no
Elevator-directed [11]	yes	no	yes	no	no	no	no	no	no	TO	no	no
Filter Collaboration [50]	yes	yes	yes	yes	yes	yes	no	yes	yes	no	no	yes
Four Player Game [36]	yes	yes	yes	no	no	yes	no	yes	no	yes	no	yes
Health System [37]	yes	yes	yes	yes	yes	no	no	yes	yes	no	no	yes
Logistic [41]	yes	yes	yes	yes	yes	yes	no	yes	no	no	no	yes
Sanitary Agency (mod) [44]	yes	yes	yes	yes	yes	no	no	yes	yes	TO	no	yes
TPM Contract [28]	yes	yes	yes	no	yes	yes	no	yes	no	no	no	no
2-Paxos 2P3A (App F)	yes	yes	yes	yes	yes	yes	yes	yes	yes	no	no	yes
Promela I* [18]	yes	no	yes	no	yes	yes	no	yes	yes	yes	yes	yes
Web Services* [18]	yes	yes	yes	yes	yes	yes	no	yes	yes	no	no	yes
Trade System* [18]	yes	yes	yes	yes	yes	yes	no	yes	yes	no	no	yes
Online Stock Broker* [18]	no	no	no	no	no	no	no	yes	no	no	no	yes
FTP* [18]	yes	yes	yes	yes	yes	yes	no	yes	no	no	no	yes
Client-server* [18]	yes	yes	yes	yes	yes	yes	no	yes	yes	no	no	yes
Mars Explosion* [18]	yes	yes	yes	yes	yes	no	no	yes	no	no	no	yes
Online Computer Sale* [18]	no	yes	no	yes	yes	yes	no	yes	no	no	no	yes
e-Museum* [18]	yes	yes	yes	no	yes	no	no	yes	yes	no	no	yes
Vending Machine* [18]	yes	yes	yes	yes	yes	yes	no	yes	yes	no	no	yes
Bug Report* [18]	yes	yes	yes	no	yes	yes	no	yes	no	no	no	yes
Sanitary Agency* [18]	no	yes	no	yes	yes	yes	no	yes	yes	no	no	yes
SSH* [18]	no	yes	no	yes	yes	yes	no	yes	yes	yes	no	yes
Booking System* [18]	no	yes	no	yes	yes	yes	no	yes	yes	no	no	yes
Hand-crafted Example* [18]	no	yes	no	yes	yes	no	no	yes	yes	no	no	yes

Conclusion

To summarize:

- introduction of i-RSC and weak k-MC system
- inclusion in these subclasses is **decidable**
- translation from session types preserves the semantics