Analyzing Multiparty Interaction using Conversation Types

Luís Caires, Hugo Torres Vieira
Nova - New University of Lisbon
Motivation

• Software systems often rely on the collaboration between multiple parties to realize their tasks e.g., web-service applications

How can we ensure protocol safety and progress in such a decentralized and dynamic setting?

• Sessions [Honda93, Honda et al.98] have been widely used to model typeful binary interaction

How can we extend classical sessions so as to address dynamic multiparty interaction?
Sessions and Conversations

- **Session type theory**
  
  Systems modeled in the $\pi$-calculus

  Types describe the behavior of a single participant
  ($\approx$ local types [HondaYoshidaCarbone07-08])

  progress analysis based on well-founded ordering of channels
  [Dezani et al.07]

- **Conversation type theory (this talk)**

  Systems modeled in the $\pi$-calculus extended with labels
  (to support distinguished interaction in a single medium)

  Types describe the behavior of a subset of participants
  (mixing global and local type specifications)

  progress analysis based on well-founded ordering of events
  and its propagation in communication
\( \pi \)-calculus + labels
\[ P :: = \begin{align*} & 0 \quad \text{(Inaction)} \\ & P | Q \quad \text{(Parallel Composition)} \\ & (\nu a) \ P \quad \text{(Name Restriction)} \\ & \text{rec} \ \chi.\ P \quad \text{(Recursion)} \\ & \chi \quad \text{(Variable)} \\ & \Sigma_{i \in I} \alpha_i.\ P_i \quad \text{(Prefix Guarded Choice)} \end{align*} \]

\[ \alpha :: = \begin{align*} & n \cdot \text{label}\? (x_1, \ldots, x_k) \quad \text{(Input)} \\ & n \cdot \text{label}\! (n_1, \ldots, n_k) \quad \text{(Output)} \end{align*} \]
The eChair System
The eChair System

- author
- eChair
- editor

- submit
- paper
- reject
- accept

- branch

- publish
- file

- copyright
(vchat)
(eChair • submit!(chat).
   chat • paper!(pdf).
   (chat • reject?)
   +
   chat • accept?(). chat • copyright!()))

| *eChair • submit?(x).
  x • paper?(pdf).
  (x • reject!)
  +
  x • accept!().
  editor • publish!(x). x • file!(pdf))

| *editor • publish?(y).
  y • file?(pdf). y • copyright?()
(vchat)
(eChair • submit!(chat).
  chat • paper!(pdf).
  (chat • reject?() +
   chat • accept?(). chat • copyright!()))
|
*eChair • submit?(x).
  x • paper?(pdf).
  (x • reject() +
   x • accept!).
   editor • publish!(x). x • file!(pdf))
|
*editor • publish?(y).
  y • file?(pdf). y • copyright?()
\((vchat)\)
\((eChair \cdot submit!(chat)).\)
  \(chat \cdot paper!(pdf).\)
  \((chat \cdot reject?)\)
  +
  \(chat \cdot accept?().\)
\(chat \cdot copyright!()\)
\(\)\n\(*eChair \cdot submit?(x).\)
  \(chat \cdot paper?(pdf).\)
  \((chat \cdot reject!)\)
  +
  \(chat \cdot accept!().\)
  \(editor \cdot publish!(chat).\)
  \(chat \cdot file!(pdf))\)
\(\)
\(*editor \cdot publish?(y).\)
  \(y \cdot file?(pdf).\)
  \(y \cdot copyright?)\)
(\textit{vchat})
(\textit{eChair} • submit!(\textit{chat}).
  \textit{chat} • paper!(\textit{pdf}).
  (\textit{chat} • reject?())
  +
  \textit{chat} • accept?(). \textit{chat} • copyright!()
)

\textit{eChair} • submit?(\textit{x}).
\textit{chat} • paper?(\textit{pdf}).
(\textit{chat} • reject!())
+  \textit{chat} • accept!().
  \textit{editor} • publish!(\textit{chat}). \textit{chat} • file!(\textit{pdf}))

\textit{editor} • publish?(\textit{y}).
\textit{y} • file?(\textit{pdf}). \textit{y} • copyright?()
(vchat)
(eChair • submit!(chat).
  chat • paper!(pdf).
  (chat • reject?() +
   chat • accept?(). chat • copyright!()))

| *eChair • submit?(x).
  chat • paper?(pdf).
  (chat • reject() +
   chat • accept!).
   editor • publish!(chat. chat • file!(pdf)))

| *editor • publish?(y).
  y • file?(pdf). y • copyright?()
(vchat)
(eChair • submit!(chat).
  chat • paper!(pdf).
  (chat • reject?)
  +
  chat • accept?(). chat • copyright!() )

* eChair • submit?(x).
  chat • paper?(pdf).
  (chat • reject?)
  +
  chat • accept!().
  editor • publish!(chat). chat • file!(pdf))

* editor • publish?(y).
  y • file?(pdf). y • copyright?()
(\texttt{vchat})
(\texttt{eChair • submit!(chat)}).
\texttt{chat • paper!(pdf)}.
(\texttt{chat • reject?()})
+
\texttt{chat • accept?()}. \texttt{chat • copyright!()}

| *\texttt{eChair • submit?(x)}.
\texttt{chat • paper?(pdf)}.
(\texttt{chat • reject!()})
+
\texttt{chat • accept!()}.\texttt{editor • publish!(chat). chat • file!(pdf)}

| *\texttt{editor • publish?(y)}.
\texttt{chat • file?(pdf)}. \texttt{chat • copyright?()})
eChair System Run

(vchat)
(ecChair • submit!(chat).
  chat • paper!(pdf).
  (chat • reject?)
  +
  chat • accept?(). chat • copyright!()}

| *eChair • submit?(x).
  chat • paper?(pdf).
  (chat • reject!)
  +
  chat • accept!().
  editor • publish!(chat). chat • file!(pdf)

| *editor • publish?(y).
  chat • file?(pdf). chat • copyright?())
(vchat)
(eChair • submit!(chat).
   chat • paper!(pdf).
   (chat • reject?())
   +
   chat • accept?(). chat • copyright!())

*eChair • submit?(x).
   chat • paper?(pdf).
   (chat • reject!())
   +
   chat • accept!().
   editor • publish!(chat). chat • file!(pdf)

*editor • publish?(y).
   chat • file?(pdf). chat • copyright?())
Conversation Types
Conversation Types

• Typing judgement

\[ P :: n_1:B_1 \mid n_2:B_2 \mid ... \mid n_k:B_k \]

says \( P \) interacts in \( n_i \) accordingly to the \( B_i \) spec

• Behavioral types (\( B \)) extend session types:

\[ B :: = B_1 \mid B_2 \mid \emptyset \mid \text{rec } \chi.B \mid \chi \]
\[ \mid \&_{i \in I}\{M_i.B_i\} \mid \oplus_{i \in I}\{M_i.B_i\} \]

Message types (\( M \)) are labeled and describe both external and internal message exchanges

\[ M :: = p \, \text{label}(B) \quad p :: = ! \mid ? \mid \tau \]
Typing *chat* Conversation
Typing *chat* Conversation

```
chat:
\tau \text{ paper}(T_{\text{pdf}}).
\oplus \{ \tau \text{ reject}();
    \tau \text{ accept}().
    \tau \text{ file}(T_{\text{pdf}}).
    \tau \text{ copyright}() \}
```
Typing chat Conversation

chat:
\[ \tau \text{ paper}(T_{pdf}). \]
\[ \oplus \{ \tau \text{ reject}(); \]
\[ \tau \text{ accept}(). \]
\[ \tau \text{ file}(T_{pdf}). \]
\[ \tau \text{ copyright()} \} \]
Typing chat Conversation

chat:
\[ \tau \text{ paper}(T_{pdf}). \]
\[ \oplus \{ \tau \text{ reject}(); \]
\[ \quad \tau \text{ accept}(). \]
\[ \quad \tau \text{ file}(T_{pdf}). \]
\[ \tau \text{ copyright}() \} \]
Typing *chat* Conversation

**chat:**

\[ \tau \text{ paper}(T_{\text{pdf}}). \]

\[ \oplus \{ \tau \text{ reject}(); \]

\[ \tau \text{ accept}(). \]

\[ \tau \text{ file}(T_{\text{pdf}}). \]

\[ \tau \text{ copyright}() \} \]

\[ \text{publish}(B_{\text{ed}}) \]

\[ \text{file} \]

\[ ? \text{ file}(T_{\text{pdf}}). \]

\[ ? \text{ copyright}() \} B_{\text{ed}} \]
Typing *chat* Conversation

**chat:**

\[ \tau \text{ paper}(T_{pdf}) \]

\[ \oplus \{ \tau \text{ reject}(); \]

\[ \tau \text{ accept}(). \]

\[ \tau \text{ file}(T_{pdf}) \]

\[ \tau \text{ copyright}() \}

\[ \text{copyright} \]

\[ \text{copyright} \]

\[ \text{publish}(B_{ed}) \]

\[ \text{file} \]

\[ \text{file}(T_{pdf}) \]

\[ ? \text{ file}(T_{pdf}). \]

\[ ? \text{ copyright}() \]
Typing *chat* Conversation

**chat:**

\[\tau \text{ paper} (T_{\text{pdf}}). \]
\[\oplus \{ \tau \text{ reject}(); \]
\[\tau \text{ accept}(). \]
\[\tau \text{ file} (T_{\text{pdf}}). \]
\[\tau \text{ copyright}() \}\]

\[\tau \text{ file} (T_{\text{pdf}}). \]
\[! \text{ file} (T_{\text{pdf}}) \times ? \text{ file} (T_{\text{pdf}}). \]

? copyright()
Typing \textit{chat} Conversation

\[\text{chat:}\]
\[\tau \text{ paper}(T_{pdf}).\]
\[\oplus \{ \tau \text{ reject();}\]
\[\quad \tau \text{ accept().}\]
\[\quad \tau \text{ file}(T_{pdf}).\]
\[\tau \text{ copyright() }\} \]

\[\tau \text{ file}(T_{pdf}).\]
\[? \text{ copyright() }\]
Typing chat Conversation

chat:
τ paper(T_{pdf}).
⊕ { τ reject();
    τ accept().
    τ file(T_{pdf}).
    τ copyright() }

⊕ { ! reject();
    ! accept().
    τ file(T_{pdf}).
    ? copyright() }
Typing chat Conversation

chat:
\[\tau \text{ paper}(T_{pdf}).\]
\[\oplus \{ \tau \text{ reject();} \]
\[\tau \text{ accept().} \]
\[\tau \text{ file}(T_{pdf}). \]
\[\tau \text{ copyright() } \}\]
Typing chat Conversation

\[
\begin{align*}
\text{chat:} & \\
\tau & \text{ paper(T}\_\text{pdf}). \\
\oplus & \{ \tau \text{ reject();} \\
\tau & \text{ accept().} \\
\tau & \text{ file(T}\_\text{pdf}). \\
\tau & \text{ copyright()} \}
\end{align*}
\]
Typing *chat* Conversation

\[
\begin{align*}
\text{chat:} & \quad \tau \text{ paper}(T_{\text{pdf}}). \\
& \quad \oplus \{ \tau \text{ reject();} \\
& \quad \tau \text{ accept().} \\
& \quad \tau \text{ file}(T_{\text{pdf}}). \\
& \quad \tau \text{ copyright() } \}
\end{align*}
\]

\[
\begin{align*}
\text{submit}(B_{\text{ee}}) & \quad \text{paper} \\
\text{copyright} & \quad \text{file} \\
\text{publish}(B_{\text{ed}}) & \quad \text{branch} \\
\end{align*}
\]

\[
\begin{align*}
? \text{ paper}(T_{\text{pdf}}). \\
& \quad \oplus \{ ! \text{ reject();} \\
& \quad ! \text{ accept().} \\
& \quad \tau \text{ file}(T_{\text{pdf}}). \\
& \quad ? \text{ copyright() } \}
\end{align*}
\]

\[
\begin{align*}
? \text{ copyright() } & \quad \text{? copyright() } \\
B_{\text{au}} & \quad B_{\text{au}} \\
\end{align*}
\]

\[
\begin{align*}
! \text{ paper}(T_{\text{pdf}}). & \quad \oplus \{ ? \text{ reject();} \\
& \quad ? \text{ accept().}! \text{ copyright() } \}
\end{align*}
\]
Typing chat Conversation

chat:
\[ \tau \text{ paper}(T_{pdf}). \]
\[ \oplus \{ \tau \text{ reject}(); \]
\[ \tau \text{ accept}(). \]
\[ \tau \text{ file}(T_{pdf}). \]
\[ \tau \text{ copyright}() \} \]

= \[ ! \text{ paper}(T_{pdf}). \oplus \{ ? \text{ reject}(); \]
\[ ? \text{ accept}(). ! \text{ copyright}() \} \]

\[ ? \text{ paper}(T_{pdf}). \]
\[ \oplus \{ ! \text{ reject}(); \]
\[ ! \text{ accept}(). \]
\[ \tau \text{ file}(T_{pdf}). \]
\[ ? \text{ copyright}() \} \]
Typing *chat* Conversation

\[
\text{chat:} \quad \begin{cases} 
\tau \text{ paper}(T_{pdf}). \\
\oplus \{ \tau \text{ reject}(); \\
\tau \text{ accept}(). \\
\tau \text{ file}(T_{pdf}). \\
\tau \text{ copyright}() \} 
\end{cases}
\]

\[
= B_{au} \otimes B_{ec} \otimes B_{ed}
\]
Typing eChair System

(vchat)
(eChair • submit!(chat).
  chat • paper!(pdf).
  (chat • reject?())
  +
  chat • accept?(). chat • copyright!()))

| eChair • submit?(x).
  x • paper?(pdf).
  (x • reject!) (x • accept!).
  editor • publish!(x). x • file!(pdf))

| editor • publish?(y).
  y • file?(pdf). y • copyright?

::
eChair : τ submit(B_{ee}) | *? submit(B_{ee}) |
editor : τ publish(B_{ed}) | *? publish(B_{ed})
Results
Results

**Theorem (Subject Reduction)**
Let $P$ be a well-typed process, $P :: T$.
If $P \rightarrow Q$ then there is $T'$ such that $T \rightarrow T'$ and $Q :: T'$.

**Proposition (Error Freeness)**
Let $P$ be a well-typed process. Then $P$ is not an error:
- $P$ has no communication errors;
- $P$ has no illegal message races

**Corollary (Type Safety)**
Let $P$ be a well-typed process. If $P \rightarrow^* Q$ then $Q$ is not an error.

**Corollary (Conversation Fidelity)**
Let $P$ be a well-typed process, $P :: T$.
Then all conversations in $P$ follow the protocols prescribed by $T$. 
Proving Progress of Conversations
Proving Progress of Conversations

- We complement conversation typing with a proof system to ensure deadlock absence.

As traditional methods (Lynch80, Kobayashi06, Dezani et al.07) we rely on imposing an ordering on **events**.

- **Judgement** $\Gamma;\Delta \vdash P$

Events in $P$ follow a well-founded order determined by $\Gamma;\Delta$.

- **Events** $(\text{channel.label.}(x)\Gamma)$ are synchronizations in labeled channels passing channel references

Each event has associated the ordering admissible $(x)\Gamma$ for the channel which is to be passed in the message

received/sent channels must comply with the prescribed order
Ordering eChair System Events
Ordering eChair System Events

![Diagram of eChair System Events]

- **Submit**
  - **Paper**
    - **Reject**
    - **Accept**
      - **Branch**
        - **Publish**
          - **File**
    - **Copyright**

Roles:
- **Author**
- **eChair**
- **Editor**
Ordering eChair System Events

- author
- eChair
- editor

- submit
- paper
- reject
- accept
- branch
- publish
- file
- copyright
Ordering eChair System Events

- Submit
- Paper
- Reject
- Accept
- Publish
- File
- Copyright

author → eChair → editor

branch

y.file < y.copyright
Ordering eChair System Events

Diagram showing the flow of events between

- author
- eChair
- editor

Events include:
- submit
- paper
- reject
- accept
- publish

Branch points and file operations:
- file
- y.file
- y.copyright

Symbols and notations:
- \( \Gamma_{ed} \)
- \( \Gamma_{ed} \)
Ordering eChair System Events

author  eChair  editor

submit
paper
reject
accept
\{branch\}
publish.(y)_{\Gamma_{ed}}
file
copyright

x.paper < x.accept < x.reject < editor.publish.(y)_{\Gamma_{ed}} < x.file < y.copyright
Ordering eChair System Events

x.paper < x.accept < x.reject < editor.publish.(y)Γ_{ed} < x.file < x.copyright
Ordering eChair System Events

\[
\text{submit.}(x) \Gamma_{ee} \\
\text{paper} \rightarrow \text{eChair} \\
\text{reject} \leftarrow \text{eChair} \\
\text{accept} \rightarrow \text{eChair} \\
\text{copyright} \leftarrow \text{eChair}
\]

\[
\Gamma_{ee} \\
x.\text{paper} < x.\text{accept} < x.\text{reject} < \text{editor}.\text{publish.}(y) \Gamma_{ed} < x.\text{file} < x.\text{copyright}
\]
Ordering eChair System Events

```
< eChair.submit.(x)Γ_{ee} < chat.paper < chat.accept chat.reject < editor.publish.(y)Γ_{ed} < chat.file < chat.copyright
```
Ordering eChair System Events

\[\text{eChair.submit.}(x)\Gamma_{ee} \prec \text{chat.paper} \prec \text{chat.accept} \prec \text{chat.reject} \prec \text{editor.publish.}(y)\Gamma_{ed} \prec \text{chat.file} \prec \text{chat.copyright}\]
Typing eChair System

```
eChair.submit.(x)Γ_{ee} < editor.publish.(y)Γ_{ed}

⊢
(vchat)
(eChair • submit!(chat).
  chat • paper!(pdf).
  (chat • reject?)
  +
  chat • accept?(). chat • copyright!(()))

⊢
eChair • submit?(x).
  x • paper?(pdf).
  (x • reject!())
  +
  x • accept!().
    editor • publish!(x). x • file!(pdf))

⊢
editor • publish?(y).
  y • file?(pdf). y • copyright?()
```
Results
Results

Theorem (Preservation of Event Ordering)
Let $P$ be well-formed and $\Gamma;\Delta \vdash P$. If $P \rightarrow Q$ then $\Gamma';\Delta,\Delta' \vdash Q$.

Theorem (Lock Freeness)
Let $P$ be a process s.t. $P :: T$ and $\Gamma;\Delta \vdash P$. If $\text{closed}(T)$ and $P$ is not a finished process then there is $Q$ such that $P \rightarrow Q$.

A type $T$ is closed if (roughly) $T = \tau(T)$.

Finished processes only exhibit shared inputs (e.g., persistent services).

Corollary (Progress)
Let $P$ be a process s.t. $P :: T$ and $\Gamma;\emptyset \vdash P$ and $\text{closed}(T)$. If $P \rightarrow^* Q$ and $Q$ is not a finished process then $Q \rightarrow R$. 

Related Work

- **Multiparty Sessions (HondaYoshidaCarbone08)**
  Created via multicast atomic service requests, each having multiple channels; support a constant number of participants; global types specify who does what.
  In our approach, a single medium supports a dynamic number of multiple participants, interacting via labeled messages.

- **Progress in Multiparty Sessions (Bettini et al.08)**
  Builds on (HondaYoshidaCarbone08) improving on the progress result;
  Does not address the interleaving of delegated sessions.

- **Dynamic Multirole Sessions (DeniéloYoshida11)**
  Multicast atomic service request, multiple session channels; Dynamic number of participants, constant number of roles.
  Our approach addresses systems with dynamic “roles”
Concluding Remarks

• Conversation types
  Simple extension of the session type notion
  Multiparty conversations in a single communication medium
  Unify local and global behavioral specs at the same level
  Unanticipated participants may join / leave a conversation, as long as the “projection invariant” is preserved
  \[ P_1 \Join P_2 \Join \ldots \Join P_k = G_\tau \]
  Progress result on systems where processes interact in multiple conversations, supporting repeated accesses to several interleaved, possibly delegated, conversations

• Ongoing work
  Analysis of role based conversation using conversation types