DbC + Multiparty session types

Hernán Melgratti

ICC University of Buenos Aires-Conicet

27 February 2020 @ Pisa

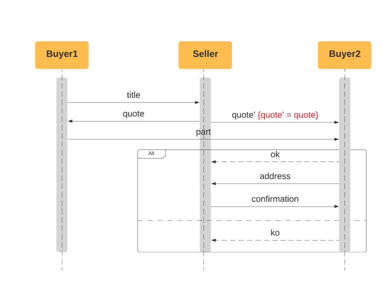
DbC + Multiparty session types[1]

- Extension of multiparty session types with assertions about communicated values

[1] Laura Bocchi, Kohei Honda, Emilio Tuosto, Nobuko Yoshida: A Theory of Design-by-Contract for Distributed Multiparty Interactions. CONCUR 2010

Global Graph (Choreography)

Two Buyers Protocol



$$
\begin{aligned}
\mathtt{TBProt} \;=\; & \mathsf{b1}\to\mathsf{s} : x\,\langle\mathtt{string}\rangle.\\
& \mathsf{s}\to\mathsf{b1} : y\,\langle \mathit{quote} : \mathsf{float}\rangle.\\
& \mathsf{s}\to\mathsf{b2} : z_1\,\langle \mathit{quote}' : \mathsf{float}\rangle\{\mathit{quote} = \mathit{quote}'\}.\\
& \mathsf{b1}\to\mathsf{b2} : z_2\,\langle\mathsf{float}\rangle.\\
& \mathsf{b2}\to\mathsf{s} : x \left\{ \begin{array}{l} \mathit{ok} : \mathsf{b2}\to\mathsf{s} : x\,\langle\mathtt{string}\rangle.\mathsf{s}\to\mathsf{b2} : z_1\,\langle\mathtt{string}\rangle.\mathsf{end},\\ \mathit{ko} : \mathsf{end} \end{array} \right\}
\end{aligned}
$$

Finite MST + Assertions

Syntax

$$
\begin{array}{rcll}
\eta & ::= & \mathsf{p} \rightarrow \mathsf{q} : x & \text{action} \\
\mathtt{G} & ::= & \eta\langle x : \tilde{\mathtt{S}}\rangle\{A\}.\mathtt{G} & \text{interaction} \\
 & \mid & \eta\{\{A_j\}l_j : \mathtt{G}_j\}_{j \in J} & \text{branch} \\
 & \mid & \mathtt{G} \mid \mathtt{G} & \text{parallel} \\
 & \mid & \mathtt{end} & \text{termination} \\
\mathtt{S} & ::= & \mathtt{int} \mid \mathtt{unit} \mid \mathtt{bool} \mid \ldots & \text{basic sorts}
\end{array}
$$

- p, r, ... : participants (also roles)
- x, y, ...: communication channels
- l, ... : labels
- $\tilde{\_}$: tuples
- A: Assertion on values

Coherence (a.k.a well-formedness)

Coherence

- $\mathtt{G}$ is coherent if it is linear and $\mathtt{G}\upharpoonright\mathtt{p}$ is well-defined for each $\mathtt{p}$
- Coherent assertions

Example

$$\mathtt{p}\rightarrow\mathtt{q} : x(v : \mathtt{int})\{v > 10\}.\mathtt{r}\rightarrow\mathtt{q} : x(w : \mathtt{int})\{w > v\}.\mathtt{end}$$

Local types + Assertions

Syntax

$$
\begin{array}{rcll}
\mathsf{T} ::= & & x?\langle v : \tilde{\mathsf{S}}\rangle\{A\}.\mathsf{T} & \text{receive} \\
 & \mid & x!\langle\tilde{\mathsf{S}}\rangle\{A\}.\mathsf{T} & \text{send} \\
 & \mid & x \oplus \{\{A_i\}l_i : \mathsf{T}_i\}_{i\in I} & \text{select} \\
 & \mid & x\&\{\{A_i\}l_i : \mathsf{T}_i\}_{i\in I} & \text{branch} \\
 & \mid & \mathtt{end} & \text{termination} \\
\\
\mathsf{S} ::= & & \mathtt{int} \mid \mathtt{unit} \mid \mathtt{bool} \mid \ldots & \text{basic sorts}
\end{array}
$$

Projection + Causal dependency on assertions

Definition

$$\begin{aligned}\mathtt{G} = \quad & \mathit{User}{\rightarrow}\mathit{Agent} : x(c : \mathit{Command})\{c \neq \mathtt{switch} - \mathtt{off}\}.\\ & \mathit{Agent}{\rightarrow}\mathit{Device} : y(c' : \mathtt{int})\{c = c'\}.\ldots\end{aligned}$$

$$\mathtt{G}{\restriction}\mathit{Agent} = y?\langle c' : \tilde{\mathtt{S}}\rangle\{c' \neq \mathtt{switch} - \mathtt{off}\}.$$

Typing

Processes $\kappa; \Gamma \vdash P \triangleright \Delta$ where κ is a constraint

$$\frac{\kappa \wedge A; \Gamma, v : \mathsf{S} \vdash P \triangleright \Delta, \tilde{s} : \mathsf{T} \mathbin{@} p}{\kappa; \Gamma \vdash s_k?(v).P \triangleright \Delta, \tilde{s} : s_k?\langle v : \mathsf{S}\rangle\{A\}.\mathsf{T} \mathbin{@} p}\ \text{Rec}$$

$$\frac{\kappa \models A\{e/v\} \qquad \Gamma \vdash \tilde{e} \triangleright \tilde{\mathsf{S}} \qquad \kappa; \Gamma \vdash P \triangleright \Delta, \tilde{y} : \mathsf{T} \mathbin{@} p}{\kappa; \Gamma \vdash s_k!v : \tilde{e}.P \triangleright \Delta, \tilde{s} : s_k!\langle v : \tilde{\mathsf{S}}\rangle\{A\}.\mathsf{T} \mathbin{@} p}\ \text{Send}$$

Property

Typing ensures that well-typed processes never violate assertions

Final words

- This is just the starting point!!! in a very active research area.
- Several works about
  - expressiveness
    - less restrictions on communication patterns (context-free, flexible merge, relaxed well-formed conditions, global graphs)
    - relaxing linearity (allowing races), shared resources
    - alternative communication models (broadcast, publish/subscribe), event notification, weak consistent logs
    - types with parameterised parties,
    - composition (open choreographies)
  - Interaction with other aspects of a language
    - Exceptions
    - Quantitative properties to reason about resource usages and complexity
    - Temporal properties
    - Probabilistic reasoning
    - Adaptability
    - Reversibility
  - Foundational aspects
    - relation with other well-known notions of programming languages (linearity, dependent types, effects)
    - Logical characterisation
    - Decomposition of Multiparty into Binary sessions
    - Synthesis (inference) of global types
    - Decidability aspects of typing/subtyping
    - Graduality
    - Monitoring

Final words

- Ensured properties
  - Type safety, Fidelity, Progress, Deadlock freedom, Lock-freedom.
  - Complete vs partial realizations
  - Security properties (e.g., information flow)
- Implementation in programming languages
  - `http://groups.inf.ed.ac.uk/abcd/session-implementations.html` (not up-to-date).
  - Typestates in Java and Join, Dependent types in Dotty (to name a few)
- New domains
  - Smart contracts