DbC for multiparty distributed interactions: static & dynamic validation

Laura Bocchi, University of Leicester
Tzu-Chun Chen, University of London
Pierre-Malo Denielou, Imperial College London
Kohei Honda, University of London
Emilio Tuosto, University of Leicester
Nobuko Yoshida, Imperial College London
**BACKGROUND**

- **DbC:** Assertions = **Types** + **Logical Formulae**

- **Type signature**

  ```
  int foobar(int i)
  ```

- **Assertion**

  ```
  int foobar(int i){
      pre: {i>10}
      post: {0< result < 1000}
  }
  ```

- **Building systems on the basis of precise contracts**
  - restrain **defensive programming**
  - provide **robustness**

---

Bertrand Meyer

*Applying “Design by Contract”*

*In Computer (IEEE), 25, 1992*
CHALLENGES

Can we extend this framework to communications and concurrency?

- **Distributed Setting** (asynchronous message passing)
- The **responsibilities** are **spread** among the participants
- Participants have **different views** of the contract, e.g., the condition of an interaction is
  - a post-condition for the sender
  - a pre-condition for the receiver
  - what about third parties?
Multiparty Session Types

Global Type

Local Type
Local Type
Local Type

Process
Process
Process

Seller $\rightarrow$ Buyer : $k_1$(Int).
Buyer $\rightarrow$ Bank : $k_2$(Int)

Buyer

$k_1?(\text{Int}).k_2!(\text{Int})$

Kohei Honda, Nobuko Yoshida and Marco Carbone
Multyparty Asynchronous Session Types (POPL 2008)
• A **global type** is used as a type signature describing the interactions of a multiparty **session**

• Each abstract action is **annotated** with a predicate
• **Consistency** of the global specification is checked
• The global assertion is **projected** onto each endpoint preserving consistency
• Each process is **validated** against its (one or more) local specification(s)
Laura Bocchi, Kohei Honda, Emilio Tuosto, and Nobuko Yoshida

A Theory of DbC for Multiparty Distributed Interactions (CONCUR 2010)

- Key points: effective well-assertedness, projection, validation*
- The proof system is sound and relatively complete

*validation is effective up to the underlying logic
Tzu-Chun Chen, Laura Bocchi, Pierre-Malo Denielou, Kohei Honda, Nobuko Yoshida

**Distributed Monitoring for Multiparty Session Enforcement**

http://www.eecs.qmul.ac.uk/~tcchen/monitoring_sessions.html

- From recent collaboration with **Ocean Observation Initiative (OOI)** on **large scale** distributed systems.
- **Unsafe** endpoints in **multiple administrative domains**.
- Use previous theory to achieve **runtime enforcement**.
Buyer $\rightarrow$ Seller : $k_1$(Int).
Seller $\rightarrow$ Buyer : $k_2\{\text{quit} : \text{End},$
\hspace{1cm} \text{ok} : \text{Buyer} \rightarrow \text{Bank} : k_3$(Int).
\hspace{1cm} \text{Bank} \rightarrow \text{Seller} : k_4$(Int) \}

GLOBAL ASSERTIONS
GLOBAL ASSERTIONS

Buyer → Seller : $k_1(o : \text{Int})$.
Seller → Buyer : $k_2\{\text{quit} : \text{End},$
\text{ok : Buyer} \rightarrow \text{Bank} : k_3(p : \text{Int})$.  
\text{Bank} \rightarrow \text{Seller} : k_4(a : \text{Int})$}
Buyer $\rightarrow$ Seller: $k_1(o : \text{Int})\{A_1\}$.  
Seller $\rightarrow$ Buyer: $k_2\{A_2\}$ quit : End,  
{true} ok : Buyer $\rightarrow$ Bank: $k_3(p : \text{Int})\{A_3\}$.  
Bank $\rightarrow$ Seller: $k_4(a : \text{Int})\{A_4\}$  

**predicates**

$A_1 = (o \geq 100)$  
$A_2 = (o < 1000)$  
$A_3 = (o = p)$  
$A_4 = (\text{true})$
When is a global assertion well designed?
“an interaction predicate can only contain those interaction variables that are known to its sender”

Alice $\rightarrow$ Bob : $(u : \text{Int})\{\text{true}\}$.

*Bob $\rightarrow$ Carol : $(v : \text{Int})\{\text{true}\}.$

Carol $\rightarrow$ Alice : $(z : \text{Int})\{z > u\}$

Alice $\rightarrow$ Bob : $(u : \text{Int})\{\text{true}\}$.

Bob $\rightarrow$ Carol : $(v : \text{Int})\{v > u\}$.

Carol $\rightarrow$ Alice : $(z : \text{Int})\{z > v\}$

Carol cannot guarantee $z > u$ since she does not know $u$. 
TEMPORAL SATISFIABILITY

“a process can always find a valid forward path at each interaction point until it meets the end”

Alice → Bob : $(v : \text{Int}) \{v > 10\}$.
Bob → Alice : $(z : \text{Int}) \{z < v \land z > 10\}$.

Alice → Bob : $(v : \text{Int}) \{v > 12\}$.
Bob → Alice : $(z : \text{Int}) \{z < v \land z > 10\}$.

- **Well-assertedness** = History Seisitvity + Temporal Satisfiability
  - is **decidable** (as long as the logic is)
  - we provide **design-time checker**

Had Alice chosen $v=11$, Carol could not find a value for $z$ s.t. $z<11$ and $z>10$.
How to project obligations and guarantees onto the endpoints?
ENDPOINT ASSERTIONS

Global assertions

\[ p \rightarrow p' : k(v : S)\{A\}.G \]
\[ p \rightarrow p' : k\{\{A_i\}l_i : G_i\}_{i \in I} \]
\[ \mu t\langle e\rangle(v : S)\{A\}.G \]
\[ t\langle e\rangle \]
\[ G, G' \]
End

Endpoint assertions

\[ k!(v : S)\{A\}; T \]
\[ k?(v : S)\{A\}; T \]
\[ k \oplus \{\{A_i\}l_i : T_i\}_{i \in I} \]
\[ k \& \{\{A_i\}l_i : T_i\}_{i \in I} \]
\[ \mu t\langle e\rangle(v : S)\{A\}.T \]
\[ t\langle e\rangle \]
End
PROJECTIONS

User $\rightarrow$ Agent : $k_1(c_1 : \text{Command})\{c_1 \neq \text{switch – off}\}$.
Agent $\rightarrow$ Instrument : $k_2(c_2 : \text{Command})\{c_2 = c_1\}$

- A too naive projection on Instrument:

$\times \quad k_2?(c_2 : \text{Command})\{c_2 = c_1\}$

$\checkmark \quad k_2?(c_2 : \text{Command})\{\exists c_1. (c_1 \neq \text{switch – off}) \land (c_2 = c_1)\}$

- We want to give stronger preconditions to prevent defensive programming
- We do not reveal the exact values exchanged between third parties
How to ensure that a process satisfies a contract expressed as an assertion?
**ASSERTED PROCESSES**

**Programs**

\[ P ::= \bar{a}[2..n]\,(\bar{s}).P \]

- request: \( a[p]\,(\bar{s}).P \)
- accept: \( (va)P \)
- hide: \( s!v\{A\};P \)
- send: \( s?v\{A\};P \)
- receive: \( s!\langle\bar{v}\rangle\{A\};P \)
- del-trw: \( s?\bar{v}\{A\};P \)

\[ s?(v)\{v \geq 10\};P \mid s:10 \cdot \bar{h} \rightarrow P[10/v] \mid s:\bar{h} \]

**Run-time processes**

\[ P_{rt} ::= P \]

- conditional: \( \text{if e then } P \text{ else } Q \)
- select: \( s < \{A\}l;P \)
- branch: \( s > \{A_i\}l_i : P_i \mid i \in I \)
- parallel: \( P \mid Q \)
- rec def: \( \mu X\langle e\rangle(v\bar{s}).P \)
- rec call: \( X\langle e\bar{s}\rangle \)
- idle: \( 0 \)

**Run-time processes**

- \( \text{errH} \) notifies a violation in a send/select
- \( \text{errT} \) notifies a violation in a receive/branch

**Receive with no violation**

\[ s?(v)\{v \geq 10\};P \mid s:10 \cdot \bar{h} \rightarrow P[10/v] \mid s:\bar{h} \]

**Receive with violation**

\[ s?(v)\{v \geq 10\};P \mid s:1 \cdot \bar{h} \rightarrow \text{errT} \mid s:\bar{h} \]
VALIDATION RULES

\[ C ::= \text{true} \mid C \land A \]  
\[ \Gamma ::= \emptyset \mid \Gamma, a : C \mid \Gamma, x : (\tilde{v} : \tilde{S}) L_1 p_1 \ldots L_n p_n \]  
\[ \Delta ::= \emptyset \mid \Delta, \tilde{s} : T @ p \]  
\[ C; \Gamma \vdash P \triangleright \Delta \]  
\[ P \text{ is validated against } \Delta \text{ and } \Gamma \]

(Rcv)

\[ C \land A; \Gamma \vdash P \triangleright \Delta, s : T @ p \]

\[ C; \Gamma \vdash s_k ?(v)\{A\}; P \triangleright \Delta, \tilde{s} : k? (v : S) \{A\}; T @ p \]

(Snd)

\[ C \subset A[e/v] \]

\[ C; \Gamma \vdash P[e/v] \triangleright \Delta, \tilde{s} : T[e/v] @ p \]

\[ C; \Gamma \vdash s_k !(e) (v)\{A\}; P \triangleright \Delta, \tilde{s} : k!(v : S) \{A\}; T @ p \]
Theorem (Soundness of Validation Rules)
Let $P$ be a closed program. Then $\Gamma \vdash P \triangleright \Delta$ implies $\Gamma \models P \triangleright \Delta$

$P$ conditionally simulates $\Delta$ and $\Gamma$
(the simulation only holds for valid inputs)

Theorem (Completeness of Validation Rules)
For each closed visible program $P$, if $\Gamma \models P \triangleright \Delta$ then $\Gamma \vdash P \triangleright \Delta$

Theorem (Error Freedom)
Let $P$ be a closed program.
Suppose
1. $\Gamma \vdash P \triangleright \Delta$,
2. $P \xrightarrow{\ell_1 \ldots \ell_n} P'$ such that $\langle \Gamma, \Delta \rangle$ allows $\ell_1 \ldots \ell_n$.
Then $P'$ contains neither $\text{err}H$ nor $\text{err}T$. 
• Error Freedom guarantees absence of violations if **ALL processes** are **validated**

• What about systems with **unsafe** endpoints? **Monitoring**!
Enabling environmental science observatories with persistent and interactive capabilities

OOI cyberinfrastructure (OOI CI) based on **loosely coupled** distributed services and agents (e.g., seafloor instruments, on-shore research stations) communicating through a **common messaging infrastructure**.

Systems are **large scale, distributed, multi-organizational**

Applications built form application-level protocols

Need for global safety ensurance by local validation with possibly unsafe endpoints

[OOI (Ocean Observatories Initiative)](http://www.oceanleadership.org/programs-and-partnerships/ocean-observing/ooi/)
INSTRUMENT COMMAND

\[ A = (y \geq 0) \]
\[ A2 = (x_n > 0) \]
\[ A4 = (x_p = \text{high} \land x_e \neq \text{busy}) \]
\[ A5 = (y > 0 \land x_{com} \neq \text{switch-off}) \]
\[ \mathcal{T} = \textbf{Buyer}!k(o : \text{Int})\{o \geq 100\}.\mathcal{T}' \]

\[ P = s_k!(o : \text{Int}).P' | s[\text{Buyer}] : \emptyset \]

\[ P_1 = P' | s[\text{Buyer}] : \langle \text{Buyer, Seller}, \langle 80 \rangle \rangle \]

\[ P_2 = P'[80/o] | s[\text{Buyer}] : \emptyset \]
PROPERTIES

- **Local/global conformance**: a monitored process well behaves and coherence is preserved in a network.

- **Local/global transparency**: monitors do not alter well-behaved interactions.

- **Session fidelity**: the interactions of a network are step-by-step conform to the corresponding global types.
Theorem (Local Conformance) $\mathcal{M} \models \mathcal{M}[P]$ for all $\mathcal{M}$ and $P$

Theorem (Global Conformance) $N \xrightarrow{\ell} N'$ with $N$ coherent implies $N'$ is coherent

Theorem (Local Transparency) If $\mathcal{M} \models \mathcal{M}^\circ[P]$ then $\mathcal{M} \models \mathcal{M}^\circ[P] \sim \mathcal{M}[P]$

Theorem (Global Transparency) Suppose $N$ is coherent and locally conformant. Then $N \sim \text{erase}(N)$

Theorem (Session Fidelity) If $\mathcal{E} \vdash N$ and $N \xrightarrow{\ell} N'$ then $\mathcal{E} \xrightarrow{\ell} \mathcal{E}'$ such that $\mathcal{E}' \vdash N'$
CONCLUSIONS

- We enabled DbC for distributed interactions through the elaboration of MPSTs with logic formulae
- Local validation of global safety
- Sound + relatively complete validation system
- Effectiveness
- Local enforcement of global safety with unsafe endpoints
- Prototype: framework for interoperable processes (Scala, Java, OCaml)
- Efficiency
RELATE WORK

HML

- M. Berger, K. Honda, and N. Yoshida. Completeness and logical full abstraction for modal logics for the typed pi-calculus. ICALP 2008


Contracts


RELATED WORK

Assertions for functional programming

DBC

Corresponding assertions, refinement/dependent types