A theory of DbC for multiparty distributed interactions

Laura Bocchi
University of Leicester

Emilio Tuosto
University of Leicester

Kohei Honda
Queen Mary, University of London

Nobuko Yoshida
Imperial College London
Aims & Objectives

- Support description/engineering of multiparty protocols
  - format of messages
  - discipline interactions in conversations
  - values carried in messages

- Devise a theoretical framework
  - analyze protocols
  - specify obligations/guarantees of participants
Reading list

**Assertion methods**

C. A. R. Hoare  
*An axiomatic basis of computer programming*  
In CACM, 12, 1969.

**Design by Contract (DbC)**

B. Meyer  
*Applying “Design by Contract”*  

**Multiparty Asynchronous Session Types**

K. Honda, N. Yoshida and M. Carbone  
*Multiparty Asynchronous Session Types*  
In POPL 2008.

**Global assertions**

L. Bocchi, K. Honda, E. Tuosto, and N. Yoshida  
*A theory of DbC for multiparty distributed*  
http://www.cs.le.ac.uk/people/lb148/assertedtypes.html
Outline

0. Assert
   Global Type
   Global Assertion
   Well-asserted Global Assertion

1. Check

2. Project
   Well-asserted $T_1$ Endpoint Assertions
   Well-asserted $T_2$ Endpoint Assertions
   Well-asserted $T_3$ Endpoint Assertions

3. Validate
   Asserted Process $P_1$
   Asserted Process $P_2$
   Asserted Process $P_3$

4. Erase
   erase($P_1$)
   erase($P_2$)
   erase($P_3$)
Design by Contract

- Type signatures to constraint computation
  - the method \( m \) of an object of class \( C \) should be invoked with a string and an integer; \( m \) will return (if ever) a string

- \( \text{DbC} = \text{Types} + \text{Assertions} \)
  - if \( m \) is invoked with a string representing a date \( 2007 \leq d \leq 2008 \) and an integer \( n \leq 1000 \) then it will (if ever) return the date \( n \) days after \( d \)

- In a distributed setting each party has
  - guarantees (e.g., on the content of the received messages)
  - obligations (e.g., on the content of the sent messages)
A simple global type

\[ \mu t(p_o : \text{int})^{100}. \]

Buyer \rightarrow Seller : ChSeller (o : int).
Seller \rightarrow Buyer : ChBuyer {
    ok: Buyer \rightarrow Bank : ChBank (p : \text{int}). Bank \rightarrow Seller : ChSeller(a : \text{bool}),
    hag: t\langle o\rangle
}

\[ t(p_o : \text{int})^{100}. \]
Outline

0. Assert

Global Type

1. Check

Global Assertion

2. Project

Well-asserted Global Assertion

Well-asserted Endpoint Assertions

Well-asserted Endpoint Assertions

Well-asserted Endpoint Assertions

3. Validate

Assisted Process P1

Assisted Process P2

Assisted Process P3

safe in untrusted env

safe in trusted env

4. Erase

erase(P1) Process

erase(P2) Process

erase(P3) Process
Syntax for Global Assertions

\[
S ::= \text{bool} \mid \text{int} \mid \ldots \mid G
\]

\[
L ::= \{p, p'\}
\]

\[
G ::= p \rightarrow p' : k (\tilde{v} : \tilde{S})\{A\}.G
\]

| \( p \rightarrow p' : k \{ \{A_j\} l_j : G_j\}_{j \in J} \) |
| \( \mu t(\ldots v_i : S_i @ L_i \ldots)\langle \tilde{e} \rangle\{A\}.G \) |
| \( t\langle \tilde{e} \rangle \) |
| \( G, G' \) |
| end |

What is A?
The investigation of the most suitable logic is left as a future work.

The logical language is likely to be application dependent.

Good candidates are first-order decidable logics (e.g., Presburger arithmetic).
A global assertion

\[ G_{\text{hag}} = \mu t(o : \text{int} @ \{\text{Buyer, Seller}\}) (10) \{ o \geq 10 \}. \]

\text{Buyer} \rightarrow \text{Seller}: \text{ChSeller} (p : \text{int}) \{ p \geq 10 \}.

\text{Seller} \rightarrow \text{Buyer}: \text{ChBuyer} \{
  \{ \text{true} \} \text{ ok:}
  \text{Buyer} \rightarrow \text{Bank}: \text{ChBank} (c : \text{int}) \{ c = p \}.
  \text{Bank} \rightarrow \text{Seller}: \text{ChSeller} (a : \text{bool}) \{ \text{true} \},
  \{ p > o \} \text{ hag: } t \langle p \rangle
\}
Correctness of Global Assertions

- Global assertions must respect 3 principles
  - History sensitivity
  - Locality
  - Temporal satisfiability
History sensitivity principle

An interaction predicate can only constraint variables known by the sender.

Alice → Bob : ChB (u:int) \{true\}.
Bob → Carol : ChC (v:int) \{true\}.
Carol → Alice : ChA (z:int) \{z<u\}

Alice → Bob : ChB (u:int) \{true\}.
Bob → Carol : ChC (v:int) \{v<u\}.
Carol → Alice : ChA (z:int) \{z<v\}

Carol cannot guarantee z<u as she doesn’t know u...

z indirectly depends on u...

...but Carol can choose the right value since she knows v and the predicates ensure that the dependencies are respected.
Locality principle

Predicates can only constraint variables which they introduce.

Alice $\rightarrow$ Bob : ChB ($u$:$\text{int}$) $\{u>0\}$.
Bob $\rightarrow$ Carol : ChC ($v$:$\text{int}$) $\{\text{true}\}$.
Carol $\rightarrow$ Alice : ChA ($z$:$\text{int}$) $\{z \geq v \land v>1\}$

Carol strengthens the constraints on $v$ without being entitled.

Alice $\rightarrow$ Bob : ChB ($u$:$\text{int}$) $\{u>0\}$.
Bob $\rightarrow$ Carol : ChC ($v$:$\text{int}$) $\{\text{true}\}$.
Carol $\rightarrow$ Alice : ChA ($z$:$\text{int}$) $\{z \geq v \land z>1\}$
Temporal-satisfiability principle

- For each value satisfying a predicate $A$
  - there is always a branch enabled
  - for each subsequent predicate $A'$, it is always possible to find values that satisfy $A'$

Alice $\rightarrow$ Bob: $\text{ChB (u:int) \{u<10\}}$.
Bob $\rightarrow$ Alice: $\text{ChA (v:int) \{v<u \land v>6\}}$.

had Alice sent 6 or 7, Bob couldn’t meet his obligation!
Being true to our principles

HSP can be statically checked
Being true to our principles

TSP implies LP, and we give a checker

\[ \text{GSat}(G, A) \]

1. \( G = p_1 \rightarrow p_2: k(\tilde{v} : \tilde{S})\{A'\}.G' \)
   \[ \begin{cases} \text{if } A \supset \exists \tilde{v}(A') \text{ then } \text{GSat}(G, A) = \text{GSat}(G', A \land A') \\ \text{otherwise } \text{GSat}(G, A) = \text{false} \end{cases} \]

2. ...

6. \( G = \text{end} \) then \( \text{GSat}(G, A) = \text{true} \)
Well-assertedness

- A global assertion $G$ is well-asserted when
  - $G$ is history-sensitive and
  - $\text{GSat}(G, \text{true}) = \text{true}$
Global Type

0. Assert

Global Assertion

1. Check

Well-asserted Global Assertion

2. Project

Well-asserted Endpoint Assertions

3. Validate

Well-asserted Endpoint Assertions

4. Erase

Well-asserted Endpoint Assertions

Process erase(P1)

Process erase(P2)

Process erase(P3)

safe in untrusted env

safe in trusted env
End-point assertions

Endpoint assertions specify the behaviour of processes involved in a session.
Projections & “third parties”

Seller $\rightarrow$ Buyer : $\text{ChBuyer}(p : \text{int}) \{p > 10\}$.  
Buyer $\rightarrow$ Bank : $\text{ChBank} (c : \text{int}) \{c \geq p\}$

A too naive projection wrt Bank would give

$\text{ChBank?}(c:\text{Int}) \{c \geq p\}$

which is meaningless because Bank ignores the value of p so it cannot check if Buyer meets its obligation.
Projecting global assertions

\[ Proj(p_1 \rightarrow p_2 : k (\tilde{v} : \tilde{S})\{A\}.G', A_{Proj}, p) = \]
\[ \begin{cases} 
  k!(\tilde{v} : \tilde{S})\{A\}.G_{Proj} & \text{if } p = p_1 \\
  k?(\tilde{v} : \tilde{S})\{\exists V_{ext}(A \land A_{Proj})\}.G_{Proj} & \text{if } p = p_2 \\
  G_{Proj} & \text{otw} 
\end{cases} \]

\[ G_{Proj} = Proj(G', A \land A_{Proj}, p) \text{ and } V_{ext} = \text{var}(A_{Proj}) \setminus T(G) \upharpoonright p \]

Seller → Buyer : \( \text{ChBuyer}(p : \text{int}) \{ p > 10 \} \).

Buyer → Bank : \( \text{ChBank}(c : \text{int}) \{ c \geq p \} \)

ChBank?(c:Int) \( \{ \exists p. \ p \geq 10 \land c \geq p \} \)

projected wrt Bank
Projection in action

\[ T_{\text{hag}} = \mu t(o : \text{int}) \langle 10 \rangle \{ o \geq 10 \}. \]

\[ \text{ChSeller?(p : \text{int})} \{ p \geq 10 \land o \geq 10 \}; \]

\[ \text{ChBuyer} \oplus \{ \]

\[ \{ \text{true} \} \text{ ok: ChSeller?(a : \text{bool})} \{ \exists c. p \geq 10 \land o \geq 10 \land c = p \} \]

\[ \{ p > o \} \text{ hag: } t\langle o \rangle, \]

\[ } \]

\[ G_{\text{hag}} = \mu t(o : \text{int}@\{\text{Buyer, Seller}\}) \langle 10 \rangle \{ o \geq 10 \}. \]

\[ \text{Buyer } \rightarrow \text{ Seller: ChSeller (p : \text{int})} \{ p \geq 10 \}. \]

\[ \text{Seller } \rightarrow \text{ Buyer: ChBuyer}\{ \]

\[ \{ \text{true} \} \text{ ok: Buyer } \rightarrow \text{ Bank: ChBank (c : \text{int})} \{ c = p \}. \]

\[ \text{Bank } \rightarrow \text{ Seller: ChSeller (a : \text{bool})} \{ \text{true} \}, \]

\[ \{ p > o \} \text{ hag: } t\langle p \rangle \]

\[ } \]
Well-assertedness for endpoint assertions

Similarly to global assertions, we define $\text{LSat}(T, A)$ to check if $T$ is satisfiable under assertion $A$. Notice that for endpoint assertions only $\text{TSP}$ is important as $\text{TSP}$ implies $\text{LP}$, $\text{HSP}$ is vacuously guaranteed, and $\text{Proj}(G, \text{true}, p)$ preserves well-assertedness.
Asserted processes

\[ P ::= \overline{a}[2..n] (\tilde{s}).P \quad \text{request} \]
\[ | a[p] (\tilde{s}).P \quad \text{accept} \]
\[ | s!(\tilde{e})(\tilde{v})\{A\}; P \quad \text{send} \]
\[ | s?(\tilde{v})\{A\}; P \quad \text{reception} \]
\[ | s < \{A\}l; P \quad \text{select} \]
\[ | s \triangleright \{\{A_i\}l_i : P_i\}_{i \in I} \quad \text{branch} \]

\[ D ::= \{ (X_i(\tilde{v}_i\tilde{s}_i) = P_i) \}_{i \in I} \quad \text{rec def} \]
\[ e ::= n \mid e \land e' \mid \neg e \ldots \quad \text{expressions} \]

\[ n ::= a \mid \text{true} \mid \text{false} \quad \text{values} \]

\[ \text{conditional} \]
\[ \text{error} \]
\[ \text{parallel} \]
\[ \text{idle} \]
\[ \text{hiding} \]
\[ \text{rec def/call} \]
Semantics

\[ a_{[2..n]}(s).P \mid a_{[2]}(s).P_2 \mid \ldots \mid a_{[n]}(s).P_n \rightarrow (\nu s)(P_1 \mid P_2 \mid \ldots \mid P_n \mid s_1:\emptyset \mid \ldots \mid s_n:\emptyset) \]

\[ s!(\bar{c})(\bar{v})\{A\}; P \mid s:\tilde{h} \rightarrow P[\bar{\nu}/\bar{v}] \mid s_k:\tilde{h} \cdot \bar{\nu} \quad (\bar{c} \downarrow \bar{\nu} \land A[\bar{\nu}/\bar{v}] \downarrow \text{true}) \]

\[ s?(\bar{v})\{A\}; P \mid s:\tilde{n} \cdot \tilde{h} \rightarrow P[\bar{\nu}/\bar{v}] \mid s:\tilde{h} \quad (A[\bar{\nu}/\bar{v}] \downarrow \text{true}) \]

\[ s \triangleright \{\{A_i\}l_i: P_i\}_{i \in I} \mid s: l_j \cdot \tilde{h} \rightarrow P_j \mid s:\tilde{h} \quad (j \in I \land A_j \downarrow \text{true}) \]

\[ s \triangleleft \{A\}l; P \mid s:\tilde{h} \rightarrow P \mid s:\tilde{h} \cdot l \quad (A \downarrow \text{true}) \]

if \( e \) then \( P \) else \( Q \rightarrow P \quad (e \downarrow \text{true}) \) if \( e \) then \( P \) else \( Q \rightarrow Q \quad (e \downarrow \text{false}) \)

\[ \text{def } D \text{ in } C[X\langle\tilde{v}\rangle] \rightarrow \text{def } D \text{ in } Q(\text{where } \langle X\langle\tilde{v}\rangle = P \rangle \in D \text{ and } C[P[\bar{\nu}/\bar{v}] \rightarrow Q}) \]

\[ s!(\bar{n})(\bar{v})\{A\}; P \rightarrow \text{errH} \quad (A[\bar{\nu}/\bar{v}] \downarrow \text{false}) \]

\[ s?(\bar{v})\{A\}; P \mid s:\tilde{n} \cdot \tilde{h} \rightarrow \text{errT} \mid s:\tilde{h} \quad (A[\bar{\nu}/\bar{v}] \downarrow \text{false}) \]

\[ s \triangleright \{\{A_i\}l_i: P_i\}_{i \in I} \mid s: l_j \cdot \tilde{h} \rightarrow \text{errT} \mid s:\tilde{h} \quad (j \in I \land A_j \downarrow \text{false}) \]

\[ s \triangleleft \{A\}l; P \rightarrow \text{errH} \quad (A \downarrow \text{false}) \]
Outline

0. Assert
- Global Type
- Global Assertion
  - Well-asserted Global Assertion

1. Check
- Well-asserted Global Assertion

2. Project
- Well-asserted T₁ Endpoint Assertions
- Well-asserted T₂ Endpoint Assertions
- Well-asserted T₃ Endpoint Assertions

3. Validate
- Asserted Process P₁
- Asserted Process P₂
- Asserted Process P₃

4. Erase
- erase(P₁) Process
- erase(P₂) Process
- erase(P₃) Process

safe in untrusted env
safe in trusted env
Validating asserted processes

\[ C; \Gamma \vdash P \triangleright \Delta \]

under the assertion environment \( C \) and the sorting \( \Gamma \), \( P \) is validated w.r.t. the assertion assignment \( \Delta \)

\[
\begin{align*}
C \supset A[\tilde{e}/\tilde{v}] & \quad C; \Gamma \vdash P[\tilde{e}/\tilde{v}] \triangleright \Delta, \tilde{s}: T[\tilde{e}/\tilde{v}] \at \mathbf{p} \\
C; \Gamma \vdash s_k!(\tilde{e})\langle \tilde{v} \rangle \{A\}; P \triangleright \Delta, \tilde{s}: k!(\tilde{v} : \tilde{S})\{A\}; T \at \mathbf{p} \\
C \land A_i, \Gamma \vdash P_i \triangleright \Delta, \tilde{s}: T_i \at \mathbf{p} & \quad \forall i \in I \\
C; \Gamma \vdash s_k \triangleright \{\{A_i\}l_i : P_i\}_{i \in I} \triangleright \Delta, \tilde{s}: k\&\{\{A_i\}l_i : T_i\}_{i \in I} \at \mathbf{p}
\end{align*}
\]
Main Results

- Validated processes can be “simulated” by their end-point assertions.
- End-point assertions do not yield errors (by construction).
- A validated process never reaches errors.
- Validation is decidable.
Main results 2

- In a trusted environment, all assertions of validate processes can be turned into true.
- A monitor can be automatically deduced from validated processes (it has to check sent/selection messages).
- In an untrusted environment, the monitor may guard processes and help in debugging.
Future work

- Study properties of suitable logics for global assertions
  - tractability/decidability
  - complexity
- Play with implementations
- Apply this context to financial protocols
Thank you...