Multiparty Session Types

BETTY Summer School 2016

Laura Bocchi, University of Kent, l.bocchi@kent.ac.uk
Agenda

- MPSTs extensions
- Design by Contract
- Beyond static typing
- Relationship with automata
- Time
- Realizability
MPSTs extensions

- Structure of interactions
  - Symmetric sum types
    [Nielsen,Yoshida,Honda@Express’10][Henriksen et al.@FHIES’12]
  - Exceptions/Escapes
    [Capecchi,Giachino,Yoshida@FTTCS’10][Carbone,Yoshida,Honda@SFM’09]
  - Dynamic multirole session types
    [Denielou,Yoshida@POPL’11]
  - Parameterised sessions
    [Denielou et al.@LMCS’12]
  - Nested sessions
    [Demangeon,Honda@CONCUR’12]

- Design by Contract (assertions on message content)
  [Bocchi,Honda,Tuosto,Yoshida@CONCUR’10] [Bocchi,Demangeon,Yoshida@TGC’12]

- Time [Bocchi,Yang,Yoshida@CONCUR’14]
Design by Contract

Assertions = Types + Logical Formulae

- Type signature
  ```
  int hello(int i)
  ```

- Assertion
  ```
  int hello(int i)
  {
    pre: {i>10}
    post: {result>0}
  }
  ```

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

[Bertrand Meyer, 1992]
A theory of DbC for concurrency

User → Agent : $s_1$ (Command).
Agent → Instrument : $s_2$ (Command).
Instrument → Agent : $s_1$ (Response).
Agent → User : $s_3$ (Response)

Bocchi, Honda, Tuosto, Yoshida@CONCUR’10
“A theory of Design by Contract for Distributed Multiparty Interactions”
A theory of DbC for concurrency

User → Agent: $s_1(x_c: \text{Command})\{x_c \neq \text{reset}\}$.
Agent → Instrument: $s_2(x_{cc}: \text{Command})\{x_{cc} = x_c\}$.
Instrument → Agent: $s_1(x_r: \text{Response})$.
Agent → User: $s_3(x_{rr}: \text{Response})\{x_{rr} = x_r\}$

Bocchi, Honda, Tuosto, Yoshida@CONCUR’10
“A theory of Design by Contract for Distributed Multiparty Interactions”
A theory of DbC for concurrency

- Participants have different views of the contract
- Responsibilities spread among the participants
- e.g., a predicate associated to an interaction is
  - a pre-condition for the receiver
  - a post-condition for the sender
Rethinking realizability

• Is this a good choreography?

• No, Carol does not have enough information to fulfil her obligation
Rethinking realizability

• Is this a good choreography?

Alice

Bob

Carol

\[ x_3 > x_2 \]

\[ x_2 > x_1 \]

• Yes, Carol knows the variables in the predicate she must guarantee
Rethinking realizability

• Is this a good choreography?

• No, Bob may “find” his obligation unsatisfiable
Rethinking realizability

• Is this a good choreography?

• Yes, there is always a solution to the constraint Bob must guarantee
Rethinking realizability

- We want to prevent:
  - that a participant does not have enough information to make correct choices
  - that a participant has no other alternatives than violating the contract

**History Sensitivity:**
“*a predicate only contains variables that are known to the sender***”

**Temporal Satisfiability:**
“*a process can find a valid forward path at each interaction point***”

---

Bocchi, Lange, Tuosto@ICE’11, SACS’12
Amending Contracts for Choreographies
Projecting assertions

• Projection - build the strongest set of preconditions for each role

\[ \text{Seller} \rightarrow \text{Buyer}: k_1 (\text{cost} : \text{Int}) \{ \text{cost} > 10 \} \].\text{Buyer} \rightarrow \text{Bank}: k_2 (\text{pay} : \text{Int}) \{ \text{pay} \geq \text{cost} \}.\text{end} \]

\[ ?(\text{Buyer}, (\text{pay} : \text{Int})) \{ \text{pay} \geq \text{cost} \}.\text{end} \]

\[ \exists \text{cost}.\text{cost} > 10 \land \text{pay} \geq \text{cost} \]
Typing (some ideas...)

\[ C ; \Gamma \vdash P : \Rightarrow \Delta \]

- **typing environment**
- **assertion environment**
- **session environment**

\[
\frac{\Gamma \vdash e : S \quad C \supseteq A[e/x] \text{ valid}}{C ; \Gamma \vdash c[p]!e;P : \Rightarrow \Delta, c : T[e/x]} \quad \text{[SND]}
\]

\[
\frac{C \land A \land \Gamma, x : S \vdash P : \Rightarrow \Delta, c : T}{C ; \Gamma \vdash c[p]?x;P : \Rightarrow \Delta, c : ?p(x : S)\{A\} ; T} \quad \text{[RCV]}
\]
Properties

- **Decidability** of type-checking depends on the logics (e.g., some fragment of Presburger arithmetic)

- **Soundness**: the behaviour of the system can be mimicked by the types

- **Completeness**: if a process is well-behaved then it can be typed (for a class of processes that do not “get stuck” at a session initiation)
Static typing

If all programs are well typed then the system is well behaved.
Static typing
**Ocean Observatory Initiative (OOI)**

**OOI aims:** to deploy an infrastructure (global network) to expand the scientists’ ability to remotely study the ocean

**Usage:** Integrate real-time data acquisition, processing and data storage for ocean research,…
OOI: verification challenges

- applications written in **different** languages, running on **heterogeneous** hardware in an **asynchronous** network.
- different authentication domains, external **untrusted** applications
- various distributed protocols
- requires correct, safe interactions
Monitoring with MPST

- Monitors between trusted network and untrusted apps
  - drop violating incoming/outgoing messages
  - ensure interoperability (no access to source code)

Bocchi, Chen, Demangeon, Honda, Yoshida@FORTE’13
Monitoring Networks through Multiparty Session Types
Run-time monitoring

\[ N = [P_1]_\alpha \mid [P_2]_\beta \mid \ldots \mid M_\alpha \mid M_\beta \mid \ldots \mid \langle r, h \rangle \]

\[ \alpha : \langle \Gamma, \Delta \rangle \]

Satisfaction \( \models N : \Sigma \)

- if \( \Sigma \) expects an \textbf{input} then \( N \) should be able to process it
- if \( N \) performs an \textbf{output} then \( \Sigma \) should allow it
- still holds after reduction
Properties (1/2)

Safety

\[ \models [P]_\alpha \mid M : \alpha : \langle \Gamma, \Delta \rangle \text{ with } M = \alpha : \langle \Gamma, \Delta \rangle. \]

a monitored process satisfies its specification

If \( N \) is fully monitored w.r.t. \( \Sigma \) then \( \models N : \Sigma. \)

a monitored network satisfies its (global) specifications
Properties (2/2)

Transparency

- a well-behaved monitored process (resp. network) remains so when monitored
- It allows mixed network with statically checked processes
- Extended with time [Neykova, Bocchi, Yoshida@BEAT’14]
- Transparency may not hold in the timed scenario…why?
Agenda

- MPSTs extensions
- Design by Contract
- Beyond static typing
- Relationship with automata
- Time
- Realizability
Theorem 4.3 (soundness and completeness). Suppose $S$ is basic and multiparty compatible. Then there exists $G$ such that $S \approx G$. Conversely, if $G$ is well-formed, then there exists a basic and multiparty compatible system $S$ such that $S \approx G$.

**basic** = deterministic, directed, no mixed state

**multiparty compatible** = in all reachable stable states, all possible (I/O) action can be matched with a complementary (O/I) action of the rest of the system after some 1-bounded executions
MPST, Automata & time

Timed Multiparty Session Types
Bocchi, Yang, Yoshida@CONCUR’14
Time-sensitive choreographies

- **Protocol specification**: deadlines, timeouts, repeated constraints, …

- **Web Services**: “Reconnect no more than twice every four minutes …” [Twitter Streaming API]

- **Sensor Networks** (on busy waiting): “Main sources of energy inefficiency in Sensor Networks are collisions and listening on idle channels” [Ye, Heidemann & Estrin, 2002]
Time model

- Each role owns a local clock
- All clocks synchronised at the beginning of the session
- Thereafter time flows at the same pace for all participants
- Send/receive actions guarded by constraints on clocks
- Clocks can be reset when sending or receiving

Delays (in milliseconds)
- $L = 400$ (latency)
- $W = 300,000$ (sampling time)
- $D = 2000$ (decision time)
Timed MPST

\[ G ::= p \rightarrow q : \{ l_i(S'_i)\{A_i, A'_i\}.G_i \}_{i \in I} | \mu t.G | t | \text{end} \]

\[ \mu t. \begin{cases} M \rightarrow W : \langle \text{task} \rangle & \{ x = 0, \emptyset, \quad L \leq y \leq L + 1, y \} \end{cases} \]

\[ W \rightarrow M : \langle \text{data} \rangle \]

\[ M \rightarrow A : \{ \text{MORE} \langle \text{data} \rangle \} \]

\[ M \rightarrow W : \text{MORE} \langle \text{task} \rangle \]

\[ t, \quad \text{STOP} \langle \text{data} \rangle \]

\[ M \rightarrow W : \text{STOP} \]

\[ \text{end} \]

\[ \} \]
Separated semantics

- Actions (send, receive, “storage into queues”) take no time
- Time is modelled separately as time actions

\[
\begin{align*}
\mu t. & \quad M \rightarrow W: \langle \text{task} \rangle \\
W & \rightarrow M: \langle \text{data} \rangle \\
M & \rightarrow A: \{ \text{MORE} \langle \text{data} \rangle \} \\
M & \rightarrow W: \text{MORE} \langle \text{task} \rangle \\
\text{STOP} & \langle \text{data} \rangle \\
M & \rightarrow W: \text{STOP} \\
\end{align*}
\]

\[
\begin{align*}
\{ x = 0, \emptyset, & \quad L \leq y \leq L + 1, \; y \} . \\
\{ y \leq W, \emptyset, & \quad x = 2L + W, \; x \} . \\
\{ x \leq D, \emptyset, & \quad z \geq 3L + W + D, \; z \} . \\
\{ x \leq D, x, & \quad y = 2L + D, \; y \} . \\
\{ x \leq D, x, & \quad z \geq 3L + W + D, \emptyset \} . \\
\{ x \leq W, x, & \quad y = 2L + D, \emptyset \} .
\end{align*}
\]

\[
\begin{align*}
& x = 0 \\
& y = 0 \\
& z = 0 \\
& \text{MW! Task} \quad x = 0 \\
& y = 0 \\
& z = 0 \\
& L \quad x = L \\
& y = L \\
& z = L \\
& \text{MW? Task} \quad x = L \\
& y = 0 \\
& z = L \\
& \ldots
\end{align*}
\]
Specified semantics

\( x = 0, \ y = 0 \)

\( p \rightarrow q : \langle \text{int} \rangle \ \{ 1 \leq x \leq 10, \ \emptyset, \ y > 10, \ \emptyset \}. \ \text{end} \)

Specified semantics:

\[
pq!\langle\text{int}\rangle
\]

\[
\downarrow
\]

\[
15
\]

\[
\downarrow
\]

\[
1.99 \quad \rightarrow
\]

\[
pq!\langle\text{int}\rangle
\]
Specified semantics

\[ x = 0, \ y = 0 \]

\[ p \rightarrow q : \langle \text{int} \rangle \quad \{ 1 \leq x \leq 10, \ \emptyset, \ y > 10, \ \emptyset \} \quad \text{end} \]

Specified semantics:

\[ \begin{array}{c}
\text{pq!}\langle \text{int} \rangle \\
\times
\end{array} \]

Communication actions at times that violate the constraints are not allowed

\[ 15 \]

\[ 1.99 \quad \text{pq!}\langle \text{int} \rangle \]
Specified semantics

\[ x = 0, \quad y = 0 \]

\[ p \rightarrow q : \langle \text{int} \rangle \quad \{ 1 \leq x \leq 10, \quad \emptyset, \quad y > 10, \quad \emptyset \}. \text{ end} \]

**Specified semantics:**

- Communication actions at times that violate the constraints are not allowed
- Time actions that make constraints of some `ready` action unsatisfiable are not allowed

\[ \times \quad pq!\langle \text{int} \rangle \]

\[ 15 \]

\[ 1.99 \quad pq!\langle \text{int} \rangle \]
Rethinking realizability

• Is this a good choreography?

\[
p \rightarrow q : \langle \text{Int} \rangle \{ x > 3, \emptyset, y = 4, \emptyset \}.\text{end}
\]

\[
\begin{align*}
    x_p &= 0 \quad \rightarrow \quad 5 \quad \rightarrow \quad x_p &= 5 \\
    x_q &= 0 \quad \rightarrow \quad 5 \quad \rightarrow \quad x_q &= 5
\end{align*}
\]

• No, constraints may become unsatisfiable for \( q \) at some point
Rethinking realizability

Feasibility: “a process can find a valid forward path until it reaches the end”

• Not feasible

\[
p \rightarrow q : \langle \text{Int} \rangle \{ x > 3, \emptyset, y = 4, \emptyset \}.\text{end}
\]

• Some feasible alternatives…

\[
p \rightarrow q : \langle \text{Int} \rangle \{ x > 3 \land x \leq 4, \emptyset, y = 4, \emptyset \}.\text{end}
\]

\[
p \rightarrow q : \langle \text{Int} \rangle \{ x > 3, \emptyset, y \geq 4, \emptyset \}.\text{end}
\]
Rethinking realizability

• Is this a good choreography?

\[ p \rightarrow q : \langle \text{Int} \rangle \{ x_p < 3 \lor x_p > 3, \ x_q < 3 \lor x_q > 3 \}.G \]

• No, it may lead to inconsistent implementations

\[
P = \text{delay}(6). \ s[q]!(p,10); G \downarrow p \\
Q = s[p]?(q,x).G \downarrow q
\]

**Wait-freedom:** *The constraint of each receive action must not admit, as a solution, a time which is earlier than some solution of the corresponding send action.*
A (very) simple timed calculus

- Session pi-calculus + delay:

\[ \text{delay}(t).P \]

\[ \?M(\text{Task})\{400 \leq x_M \leq 401\}. \\
!M(\text{Data})\{x_M \leq 300401\}. \\
\ldots \]

\[ \text{delay}(400). \\
\text{s}[W]?(M, x). \\
\text{delay}(100000). \\
\text{s}[W]!(M, \text{sample}(x)). \\
\ldots \]
Typing (some ideas...)

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

typing environment

\\[ c : (\nu, T) \]

virtual clock

\[ \Gamma \vdash P \triangleright \{ c_i : (\nu_i + t, T_i) \}_{i \in I} \]

[Delay]

\[ \Gamma \vdash c[p] \triangleleft l_j(e) ; P \triangleright \Delta, c : (\nu, p \oplus \{ l_i(S_i) \{ \delta_i, \lambda_i \} . T_i \}_{i \in I}) \]

[Select]
Properties

- Verification of real-time interactions with Multiparty Session Types
- **time-error freedom**: interactions are punctual
- **time-progress**: each reachable state is not a deadlock state and time can diverge (no Zeno)

**Theorem (Time-error freedom)** If $\Gamma \vdash P \triangleright \Delta$, and $P \rightarrow^* P'$ then $P' \neq \text{error}$

**Theorem (Progress for interleaved sessions)** Let $P$ be **feasible** and **wait-free**, $\Gamma \vdash P_0 \triangleright \emptyset$, $P_0 \rightarrow^+ P$. If $P_0$ is **session delay**, **erase**$(P)$ is not a deadlock process and if **erase**$(P) \rightarrow$ then $P \rightarrow$. 

- $\times$ $\overline{a}[1].\text{delay}(6).P \mid a[1].Q$
- $\checkmark \overline{a}[1].\text{delay}(6).P \mid a[1].Q$
Theorem (Soundness and completeness) (1) Let $G$ be projectable then $\mathcal{A}(G)$ is basic and multiparty compatible. Furthermore with a specified semantics $G \approx \mathcal{A}(G')$. (2) If $C$ is basic, multiparty compatible and with a specified semantics then there exists $G$ such that $C \approx \mathcal{A}(G)$.
Theorem (Progress and liveness of CTA) If $C$ is basic, multiparty compatible and with a specified semantics and there exists a feasible $G$ s.t. $C \approx A(G)$, then $C$ satisfies progress and liveness.

- Progress: each reachable state is non-deadlock and allows time divergence
- Liveness: a final state can be reached from all reachable states

More permissive notion of feasibility, synthesis, more properties

Bocchi, Wang, Yoshida@CONCUR’14
“Timed Multiparty Session Types”

Bocchi, Lange, Yoshida@CONCUR’15
“Meeting Deadlines Together”
Wrapping up

• Practical enough for a prototype

Neykova, Bocchi, Yoshida@BEAT’14
“Timed Runtime Monitoring for Multiparty Conversations”

• Several aspects of current work uncritical for wide use
  • Separated semantics
  • Semantics closer to automata models than real world
  • Very prescriptive timing in choreographies
  • Inflexible (strictly static) definition of delays in programs
Future work

- Time-sensitive protocol design and implementation
  EPSRC - EP/N035372/1

  - expressiveness
    - set flexible schedules for the timing of actions
    - support run-time adjustments
  - tractability
  - practicality

http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/N035372/1
Agenda

- MPSTs extensions
- Design by Contract
- Beyond static typing
- Relationship with automata
- Time
- Realizability
Back to choreographies

• Are all choreographies realisable as the composition of concurrent processes?

• Choreographies and realisations are sets of traces

  \[ \ell ::= \ pq!U \mid \ pq?U \]

• Correct realisation (for now)

  \[ \text{Traces}(R) \subseteq \text{Traces}(C) \]
Realizability

- What we want: \( AB!U_1, AB?U_1, BA!U_2, BA?U_2, CA!U_3, CA?U_3 \)
Realizability

- What we want: $AB!U_1$, $AB?U_1$, $BA!U_2$, $BA?U_2$, $CA!U_3$, $CA?U_3$.
Realizability

- What we want: $\text{AB!}U_1, \text{AB?}U_1, \text{BA!}U_2, \text{BA?}U_2, \text{CA!}U_3, \text{CA?}U_3$

- What we may get: $\text{CA!}U_3, \text{AB!}U_1, \text{AB?}U_1, \text{BA!}U_2, \text{BA?}U_2, \text{CA?}U_3$
Realizability with MPST

• You cannot model the choreography that only allows trace
  \(\text{AB!} U_1, \text{AB?} U_1, \text{BA!} U_2, \text{BA?} U_2, \text{CA!} U_3, \text{CA?} U_3\)

  using global types

• E.g., \(\text{A} \rightarrow B : \langle U_1 \rangle. \text{B} \rightarrow A : \langle U_2 \rangle. \text{C} \rightarrow A : \langle U_3 \rangle. \text{end}\)

  also allows \(\text{CA!} U_3, \text{AB!} U_1, \text{AB?} U_1, \text{BA!} U_2, \text{BA?} U_2, \text{CA?} U_3\)
Realizability with MPST

• Another example, that we have already seen is:

\[
G ::= M \rightarrow W : \langle \text{task} \rangle.
    \]

\[
W \rightarrow A\{ \text{ok} : W \rightarrow A : \langle \text{data} \rangle. A \rightarrow M : \langle \text{result} \rangle. \text{end},
    \text{stop} : A \rightarrow M : \langle \text{error\_code} \rangle. \text{end} \}
\]

which is not projectable (on \(M\))

• Projectable global types model realizable choreographies:

it would be nice!

Projectable \[\rightarrow\] \(\text{Realizable}\) \[\rightarrow\] yes
References

Design by Contract

- [Bocchi et al. @ CONCUR’10] A theory of Design by Contract for Distributed Multiparty Interactions

- [Bocchi, Demangeon, Yoshida @ TGC’13] A Multiparty Multi-Session Logic. (extension with persistent variables)

- [Bocchi, Demangeon @ PLACES’13] Embedding Session Types in HML

Multiparty + Time

- [Bocchi, Yang, Yoshida @ CONCUR’14] Timed Multiparty Session Types

- [Neykova, Bocchi, Yoshida @ BEAT’14] Timed Runtime Monitoring for Multiparty Conversations

- [Bocchi, Lange, Yoshida @ CONCUR’15] Meeting Deadlines Together
References

Dynamic Monitoring

• [Bocchi et al.@FORTE’13] A theory of Design by Contract for Distributed Multiparty Interactions

• [Chen et al.@TGC’13] Monitoring Networks through Multiparty Session Types.

Automata

• [Deniélou,Yoshida@ESOP’12] Multiparty Session Types Meet Communicating Automata

• [Deniélou,Yoshida@ICALP’13] Multiparty Compatibility in Communicating Automata: Characterisation and Synthesis of Global Session Types

• [Lange,Tuosto,Yoshida@POPL’15] From communicating machines to graphical choreographies

• [Bocchi,Lange,Yoshida@CONCUR’15] Meeting Deadlines Together
References

- Mobility Reading Group’s home page [http://mrg.doc.ic.ac.uk](http://mrg.doc.ic.ac.uk)

**Binary**
- [Takeuchi,Honda,Kubo@PARLE’94] An Interaction-Based Language and its Typing System
- [Honda,Vasconcelos,Kubo@ESOP’98] Language Primitives and Type Disciplines for Structured Communication-based Programming

**Multiparty**
- [Honda,Yoshida,Carbone@POPL’08] Multiparty asynchronous session types
- [Bettini et al.@CONCUR’08] Global Progress in Dynamically Interleaved Multiparty Sessions
- [Castagna, Dezani-Ciancaglini, Padovani@FTDS’12] On Global Types and Multi-Party Sessions
- [Deniélou,Yoshida@POPL’11] Dynamic multirole session types
- [Coppo et al.@SFM’15] Gentle Introduction to Multiparty Asynchronous Session Types
References

Others

- [Deniélou, Yoshida@POPL’11] Dynamic Multirole Session Types.
- [Beljeri, Yoshida@PLACES’08] Synchronous Multiparty Session types
  - [Kouzapas, Yoshida@CONCUR’13] Globally Governed Session Semantics
  - [Honda et al@COB’12] Structuring Communication with Session Types
  - [Bocchi, Melgratti, Tuosto@ESOP’15] Resolving Non-Determinism in Choreographies

Scribble

- [Honda et al@COB’12] Structuring Communication with Session Types
- [Yoshida et al.@TGC’13] The Scribble protocol language