Sub-typing and sub-behaviour relations

Franco Barbanera\textsuperscript{1} and Ugo de’Liguoro\textsuperscript{2}

\textsuperscript{1}Dipartimento di Matematica e Informatica, Università di Catania
\textsuperscript{2}Dipartimento di Informatica, Università di Torino

WSBT - 20th April 2011, Lisboa
Session Types

Overview

Session

A *session* is a logic unit, collecting and structuring messages exchanged among a determined set of agents, sharing a private channel to prevent interference by third parties.

- **Session types** have been introduced to formalise *two-sided* sessions in type systems for the $\pi$-calculus.

We set up a behavioural semantic investigation of session types using the notion of **contract**.

- **Contracts** are a process algebraic formalism to describe the behaviour of services in a client/server scenario.
Session Types (Honda, Vasconcelos, Kubo)

Session types = regular trees of ordinary types of (polyadic) $\pi$-calculus

If $\Gamma \vdash P$ is derivable and

$$\Gamma(x) = \mu X. \ ?(\text{Int}). & \langle \ell_0 : ![\text{Bool}] \text{end},$$

$$\ell_1 : \oplus \langle \ell_2 : \text{end},$$

$$\ell_3 : X \rangle \rangle$$

then channel $x$ is used in $P$ to carry the following “session”:

1. input an integer
2. on receiving the message $\ell_0$ send a boolean then stop
3. on receiving $\ell_1$ either issue $\ell_2$ then stop, or issue $\ell_3$ and start over the whole session
Session Types (Honda, Vasconcelos, Kubo)

The syntax:

\[
T ::= \textbf{Int} \mid \textbf{Bool} \mid \ldots \mid S \quad \text{ground/session type}
\]
\[
S ::= \text{end} \quad \text{ended session}
\]
\[
\mid ?(T)S \quad \text{input of type } T, \text{ then } S
\]
\[
\mid ![T]S \quad \text{output of type } T, \text{ then } S
\]
\[
\mid \&\langle \ell_i : S_i \mid i \in I \rangle \quad \text{branching (} I \text{ finite)}
\]
\[
\mid \oplus\langle \ell_i : S_i \mid i \in I \rangle \quad \text{selection (} I \text{ finite)}
\]
\[
\mid X \quad \text{variable}
\]
\[
\mid \mu X. S \quad \text{recursion (} S \text{ not a variable)}
\]

where the \( T \) in \( ?(T) \), \( ![T] \) has to be closed (a restriction w.r.t. [HVK] and [GH] session types).

If \( T \) is restricted to ground types, these are \textit{first order} session types; they are \textit{higher-order} otherwise.
The “duality” relation over session types:

\[
\begin{align*}
\text{end} & = \text{end} \\
?T S & = ![T] \bar{S} \\
![T] S & = {?T} S \\
& \langle \ell_i : S_i \mid i \in I \rangle & = & \bigoplus \langle \ell_i : \bar{S}_i \mid i \in I \rangle \\
\bigoplus \langle \ell_i : S_i \mid i \in I \rangle & = & & \& \langle \ell_i : \bar{S}_i \mid i \in I \rangle \\
\overline{X} & = & & X \\
\mu X. S & = & & \mu X. \bar{S}
\end{align*}
\]

The following rule is at the hearth of error freeness property within a typeable session:

\[
\Delta, x : S \vdash P \quad \Delta, x : \bar{S} \vdash Q \\
\hline
\Delta \vdash (\nu x)(P|Q)
\]
Subtyping Session Types (Gay-Hole)

**Subtyping intuition**

$A <: B$ if and only if any channel that satisfies the stricter “protocol” $A$ also satisfies the protocol $B$

The $A <: B$ relation has been axiomatized by Gay and Hole.

They proved it *operationally* sound by showing that the *narrowing* rule:

$$
\Delta, x : B \vdash P \quad A <: B \\
\Delta, x : A \vdash P
$$

doesn’t break subject reduction.

Note that narrowing rule is just the dual of *subsumption* rule of the $\lambda$-calculus with subtyping.
Coinductive Axiomatization of FO-Subtyping

A coinductive reformulation: let \( \Gamma = \{ A_1 <: B_1, \ldots, A_k <: B_k \} \), then we derive judgements of the form \( \Gamma \vdash A <: B \) by the rules:

\[
\begin{align*}
\Gamma \vdash A\{\mu X. A/X\} \leq_p B & \quad \Rightarrow \quad \Gamma \vdash \mu X. A \leq_p B \\
\Gamma \vdash B \leq_p A\{\mu X. A/X\} & \quad \Rightarrow \quad \Gamma \vdash B \leq_p \mu X. A \\
\Gamma, \&_{i \in I} \langle l_i : A_i \rangle <: \&_{j \in J} \langle l_j : B_j \rangle \vdash A_i <: B_i & \quad \forall i \in I \quad I \subseteq J \\
\Gamma \vdash \&_{i \in I} \langle l_i : A_i \rangle <: \&_{j \in J} \langle l_j : B_j \rangle & \\
\Gamma, \oplus_{i \in I} \langle l_i : A_i \rangle <: \oplus_{j \in J} \langle l_j : B_j \rangle \vdash A_j <: B_j & \quad \forall j \in J \quad I \supseteq J \\
\Gamma \vdash \oplus_{i \in I} \langle l_i : A_i \rangle <: \oplus_{j \in J} \langle l_j : B_j \rangle &
\end{align*}
\]
Problem

Is there a semantic characterization of session subtyping?

Answer: behavioural semantics

- provide a formal definition of protocols as *behaviours*
- give a concept of *sub-behaviour*
- interpret session types as behaviours

We understand behaviours as a suitable kind of processes, for which we choose *contracts*
Contracts (Castagna, Laneve, Padovani)

- **contracts** are abstract specifications of web-services (and of client queries)
- central is the **compliance** relation among a client query and a server contract:

  \[ \rho \text{ complies with } \tau \quad (\rho \vdash \tau, \ \rho \text{ is a client for } \sigma) \]

  \[ \Uparrow \]

  every request from \( \rho \) is satisfied by \( \sigma \)

- compliance induces a **subcontract** relation:

  \( \sigma \) is a **subcontract** of \( \tau \) \( (\sigma \preceq \tau) \iff \) every client of \( \sigma \) is such of \( \tau \)
Contracts (Castagna, Laneve, Padovani)

Web *contracts* are parallel-free CCS terms (without $\tau$) generated by the grammar:

$$\sigma ::= 1 \mid \alpha.\sigma \mid \sigma + \sigma \mid \sigma \oplus \sigma \mid x \mid \text{rec } x.\sigma$$

where $\alpha \in \mathcal{N} \cup \overline{\mathcal{N}}$.

Semantics is defined by the LTS:

- $\alpha.\sigma \xrightarrow{\alpha} \sigma$
- $\sigma \xrightarrow{\alpha} \sigma' \Rightarrow \sigma + \rho \xrightarrow{\alpha} \sigma', \rho + \sigma \xrightarrow{\alpha} \sigma'$
- $\sigma \oplus \rho \rightarrow \sigma, \sigma \oplus \rho \rightarrow \tau$
- $\text{rec } x.\sigma \rightarrow \sigma\{\text{rec } x.\sigma/x\}$
Example

The contract of a ballot service might be:

\[
\text{rec } x. \text{Login.}(\overline{\text{Wrong}.x \oplus \overline{\text{Ok.}}. (\text{VoteA.}(Va1+Va2)+\text{VoteB.}(Vb1+Vb2)))}
\]
Example

The contract of a ballot service might be:

\( \text{rec } x. \text{Login.}(\text{Wrong}.x \oplus \text{Ok.}(\text{VoteA.}(Va1 + Va2) + \text{VoteB.}(Vb1 + Vb2))) \)

meaning:

- wait for a Login action
Example

The contract of a ballot service might be:

```
rec x. Login. (Wrong.x ⊕ Ok. (VoteA.(Va1+Va2) + VoteB.(Vb1+Vb2)))
```

meaning:

- wait for a Login action
- acknowledge the (in)correctness of login
Example

The contract of a ballot service might be:

\[
\text{rec } x. \text{Login} (\text{Wrong}.x \oplus \text{Ok}.(\text{VoteA}.(Va1 + Va2) + \text{VoteB}.(Vb1 + Vb2)))
\]

meaning:

- wait for a Login action
- acknowledge the (in)correctness of login
- in the negative restart
Example

The contract of a ballot service might be:

```
rec x. Login.(Wrong.x ⊕ Ok.(VoteA.(Va1+Va2)+VoteB.(Vb1+Vb2)))
```

meaning:

- wait for a Login action
- acknowledge the (in)correctness of login
- in the negative restart
- in the positive prompt for voting either A or B
The contract of a ballot service might be:

\[
\text{rec } x.\text{Login.} (\text{Wrong}.x \oplus \text{Ok}.(\text{VoteA}.(Va1 + Va2) + \text{VoteB}.(Vb1 + Vb2)))
\]

meaning:

- wait for a Login action
- acknowledge the (in)correctness of login
- in the negative restart
- in the positive prompt for voting either A or B
- then offer the possibility for voting for a ticket
The contract of a ballot service might be:

$$\text{rec } x. \text{Login.}(\text{Wrong}.x \oplus \text{Ok}.(\text{VoteA}.(Va1+Va2)+\text{VoteB}.(Vb1+Vb2)))$$

meaning:

- wait for a Login action
- acknowledge the (in)correctness of login
- in the negative restart
- in the positive prompt for voting either A or B
- then offer the possibility for voting for a ticket
Session Behaviours as Contracts interpreting Session Types

Consider the mapping from (first order) session types to contracts:

\[
\begin{align*}
{[X]} & = x \\
{[\text{end}]} & = 1 \\
{[\mu X. A]} & = \text{rec } x. {[A]} \\
{[\gamma(A)]} & = \gamma \cdot {[A]} \\
{[\gamma(A)]} & = \overline{\gamma} \cdot {[A]} \\
{[\&(\ell_i : B_i | i \in I)]} & = \sum_{i \in I} \ell_i \cdot {[B_i]} \\
{[\bigoplus(\ell_i : B_i | i \in I)]} & = \bigoplus_{i \in I} \overline{\ell_i} \cdot {[B_i]} \\
\end{align*}
\]

The image of the \([\cdot]\) map is a subset of the set of contracts.
Session Behaviours: the grammar

*Session Behaviours* in $S$ are the closed expressions defined by the grammar:

\[
\sigma ::= 1 \\
| a_1.\sigma_1 + \cdots + a_n.\sigma_n \quad \text{external choice, } a_i \text{ distinct} \\
| \overline{a}_1.\sigma_1 \oplus \cdots \oplus \overline{a}_n.\sigma_n \quad \text{internal choice, } \overline{a}_i \text{ distinct} \\
| x \quad \text{variable} \\
| \text{rec } x.\sigma \quad \text{recursion, } \sigma \text{ not a variable}
\]

*Contracts* describe the overall behaviour of a client or a server. *Session Behaviors* describe the possible interactions of a process over a channel.
Compliance and Orthogonality

Extend the reduction relation to pairs of session-behaviours $\rho \parallel \sigma$:

\[
\begin{align*}
\rho \xrightarrow{\alpha} \rho' & \quad \sigma \xrightarrow{\overline{\alpha}} \sigma' \\
\rho \parallel \sigma \rightarrow \rho' \parallel \sigma' \\
\rho' \parallel \sigma \rightarrow \rho' \parallel \sigma \\
\sigma \rightarrow \sigma' \\
\rho \parallel \sigma \rightarrow \rho \parallel \sigma'
\end{align*}
\]

**Compliance**: the *client* $\rho$ *complies* with the *server* $\sigma$, $\rho \vdash \sigma$ if

\[
\forall \rho', \sigma' \quad \rho \parallel \sigma \rightarrow^* \rho' \parallel \sigma' \not\rightarrow \Rightarrow \rho' = 1
\]

i.e. *any request of the client is eventually satisfied by the server.*

**Orthogonality**:

\[
\rho \perp \sigma \iff \rho \vdash \sigma \land \sigma \vdash \rho
\]
Examples

\[ \overline{a} \oplus \overline{b} \vdash a + b + c \] because:

\[
(\overline{a} \oplus \overline{b}) \| (a + b + c) \quad \rightarrow \quad \overline{a} \| (a + b + c) \quad \rightarrow \quad 1 \| 1
\]

\[
\overline{b} \| (a + b + c) \quad \rightarrow \quad 1 \| 1
\]

and also \( a + b + c \vdash \overline{a} \oplus \overline{b} \) hence \( \overline{a} \oplus \overline{b} \perp a + b + c \).

But \( \overline{a} \oplus \overline{b} \oplus \overline{c} \not\vdash a + b \) (and \( a + b \not\vdash \overline{a} \oplus \overline{b} \oplus \overline{c} \)) since:

\[
(\overline{a} \oplus \overline{b} \oplus \overline{c}) \| (a + b) \quad \rightarrow \quad \overline{c} \| (a + b) \not\rightarrow
\]

Note that \( \text{rec } x.a.x \nRightarrow \text{rec } x.\overline{a}.x \) (without reaching \( 1 \| \cdots \)) since:

\[
\text{rec } x.a.x \| \text{rec } x.\overline{a}.x \quad \xrightarrow{2} \quad a.\text{rec } x.a.x \| \overline{a}.\text{rec } x.\overline{a}.x
\]

\[
\rightarrow \quad \text{rec } x.a.x \| \text{rec } x.\overline{a}.x \rightarrow \cdots
\]
For $\sigma, \rho \in \mathcal{S}$, let

$$
\text{Client}(\sigma) = \{\rho \in \mathcal{S} \mid \rho \vdash \sigma\}, \quad \text{Server}(\rho) = \{\sigma \in \mathcal{S} \mid \rho \vdash \sigma\}
$$

Then define the relations:

1. $\sigma \preceq_s \sigma'$ if and only if $\text{Client}(\sigma) \subseteq \text{Client}(\sigma')$;
2. $\rho \preceq_c \rho'$ if and only if $\text{Server}(\rho) \subseteq \text{Server}(\rho')$.

In words: $\sigma \preceq_s \sigma'$ if the server $\sigma'$ has a larger set of clients than $\sigma$, and similarly for $\rho \preceq_c \rho'$.

Note. Our $\preceq_s$ is essentially the subcontract relation by Castagna et alii.
Let us extend the $\bar{\cdot}$ operation to all (also open) behaviours:

- $\overline{1} = 1$
- $\overline{a.\sigma} = \overline{a.\sigma}$ and $\overline{a.\sigma} = a.\overline{\sigma}$
- $\overline{\sigma + \tau} = \overline{\sigma} \oplus \overline{\tau}$
- $\overline{\sigma \oplus \tau} = \overline{\sigma} + \overline{\tau}$
- $\overline{x} = x$
- $\overline{\text{rec } x.\sigma} = \text{rec } x.\overline{\sigma}$

If $\sigma \in S$ then $\overline{\sigma} \in S$, and $\overline{\overline{\sigma}} = \sigma$. Moreover:

$$\sigma = \llbracket A \rrbracket \text{ if and only if } \overline{\sigma} = \overline{\llbracket A \rrbracket}$$
Relating the syntactic operator $\top$ to the server/client preorders:

**Proposition.** Let $\tau \in S$:

1. $\top$ is the minimum server among those of $\tau$:
   \[
   \forall \sigma \in \text{Server}(\tau). \quad \top \preceq_s \sigma \quad (\text{i.e. Client}(\top) \subseteq \text{Client}(\sigma))
   \]
2. $\top$ is the minimum client among those of $\tau$:
   \[
   \forall \rho \in \text{Client}(\tau). \quad \top \preceq_c \rho \quad (\text{i.e. Server}(\top) \subseteq \text{Server}(\rho))
   \]

This does not hold outside of $S$:

- $\overline{a} \oplus \overline{a}.b \not\models a + a.b$
- the minimum of Client($a + a.b$) is actually $\overline{a}$
- $a + a.b \not\models \overline{a} \oplus \overline{a}.b$
- the minimum of Server($a + a.b$) is $\overline{a}.b$
- Server($a.b + a.c$) = $\emptyset$
**Behavioural Subtyping**

Let $A \perp = \{ \sigma \in S \mid \exists \tau \in A. \sigma \perp \tau \}$ and $\perp = \{ \sigma \}$:

$$\sigma \preceq: \tau \iff \sigma \perp \subseteq \tau \perp$$

**Theorem**

Behavioural subtyping is the intersection of both client and server-subbehaviour relations:

$$\preceq = \preceq_c \cap \preceq_s$$

It follows that for any $\sigma, \tau \in S$, $\overline{\sigma}$ is minimal in $\sigma \perp$ w.r.t. $\preceq$: and

$$\sigma \preceq: \tau \text{ if and only if } \overline{\tau} \preceq: \overline{\sigma}$$

matching with the fact that $A <: B \iff \overline{B} <: \overline{A}$. 
**Higher-Order LTS**

**Higher-order Behaviours** add input/output of behaviors to prefixes:

\[
\sigma, \tau ::= \ldots | ?\sigma^p . \tau | !\sigma^p . \tau
\]

where \( p \in \{s, c\} \).

The higher-order LTS:

\[
\begin{align*}
?\rho^p . \sigma & \xrightarrow{\rho^p} \sigma \\
\sigma & ?\rho^p \xrightarrow{\rho^p} \sigma' \quad \tau & !\rho^p \xrightarrow{\rho^p} \tau' \quad \rho_1 \preceq_p \rho_2
\end{align*}
\]

\[
\sigma \parallel \tau \xrightarrow{} \sigma' \parallel \tau'
\]

\[
\sigma \parallel \tau \xrightarrow{} \sigma' \parallel \tau'
\]

Note the use of \( \preceq_s, \preceq_c \) in the LTS rules.

The syntactical duality extends as:

\[
?\sigma^p . \tau = !\sigma^p . \overline{\tau}, \quad !\sigma^p . \tau = ?\sigma^p . \overline{\tau}
\]
Interpreting Higher-Order Sessions

Higher-order session may send and receive session types:

\[ A, B, ::= \ldots \mid ?(A^p)B \mid ![A^p]B \quad \text{for } p = c, s \]

By considering higher-order behaviours we can extend the interpretation map to higher order session types straightforwardly:

\[
\llbracket ?(A^p)B \rrbracket = ?\llbracket A \rrbracket^p \llbracket B \rrbracket, \quad \llbracket ![A^p]B \rrbracket = ![\llbracket A \rrbracket^p \llbracket B \rrbracket
\]

Note. We have studied asymmetric session-types, with polarized channels to record either client or server role in [Barbanera-Capecci-de’Liguoro, Proc. of FSEN’09].
Subtyping Higher-Order Sessions

We decorate the sent/received session by a polarity:

\[ A, B, ::= \ldots | ?(A^p)B | !A^pB \text{ for } p = c, s. \]

Then consider the (coinductive versions of) the Gay-Hole rules:

\[
\Gamma, ?(A^p)B <: ?(C^p)D \vdash A <: C, B <: D
\]

\[
\Gamma \vdash ?(A^p)B <: ?(C^p)D
\]

\[
\Gamma, ![A^p]B <: ![C^p]D \vdash C <: A, B <: D
\]

\[
\Gamma \vdash ![A^p]B <: ![C^p]D
\]

**Fact** \( A <: B \) (according to Gay-Hole) if and only if \( \vdash A <: B \)
Main Theorem

Define:

1. $\models A <: B$ iff $[A] \trianglelefteq [B]$
2. $\models \Gamma$ iff $\models C <: D$ for all $C <: D \in \Gamma$
3. $\Gamma \models A <: B$ iff $\models \Gamma$ implies $\models A <: B$

then (soundness)

$$\Gamma \vdash A <: B \Rightarrow \Gamma \models A <: B$$

Completeness also holds:

$$\Gamma \models A <: B \Rightarrow \Gamma \vdash A <: B$$
Final Remarks

Results:

- we have proposed an interpretation of session types into behaviours which is sound w.r.t. Gay-Hole subtyping
- we also have that the interpretation is complete
- when restricting to $S$, there is no theoretical loss w.r.t. the full set of contracts in the case of two-ended sessions

Further work:

- things are different when considering multiparty sessions and fairness concepts are involved
- the power of higher-order LTS in giving semantics to the typed $\pi$-calculus deserves further attention