Processes (1)

Synchronous syntax for asynchronous communication. Let $i, n \geq 2$.

$$
\begin{array}{rcl}
P & ::= & \mathbf{0} \mid a[i](\tilde{s}).P \mid \overline{a}[2,\cdots,n](\tilde{s}).P \\
& \mid & s!\langle e\rangle;P \mid s?(x).P \\
& \mid & s \triangleleft l;P \mid s \triangleright [l_i : P_i]_{i\in I} \\
& \mid & P|Q \mid (\nu\, a)P \mid (\nu\, s_1,\cdots,s_n)P \\
& \mid & (\mu X(\tilde{x}).P)\langle\tilde{e}\rangle \mid X\langle\tilde{e}\rangle
\end{array}
$$

Processes (2)

$$\overline{a}[2..n](s_1..s_m).P_1 \mid a[2](s_1..s_m).P_2 \mid \cdots \mid a[n](s_1..s_m).P_n$$
$$\longrightarrow (\nu\,\tilde{s})(\Pi_{1\leq i\leq n}P_i \;\mid\; \Pi_{1\leq j\leq m}\, s_j : \emptyset)$$

$$s!v;P \;\mid\; s:\tilde{w} \quad\longrightarrow\quad P \;\mid\; s:v.\tilde{w}$$

$$s?(x).P \;\mid\; s:\tilde{w}.v \quad\longrightarrow\quad P[v/x] \;\mid\; s:\tilde{w}$$

$$s \triangleleft l;P \;\mid\; s:\tilde{w} \quad\longrightarrow\quad P \;\mid\; s:l.\tilde{w}$$

$$s\&[l_i.P_i]_{i\in I} \;\mid\; s:\tilde{w}.l_j \quad\longrightarrow\quad P_j \;\mid\; s:\tilde{w}$$

Assertions

- Global assertions: global types elaborated with logical formulae.
- Endpoint assertions: endpoint types elaborated with logical formulae.
- A projection function relates these two.

Global Assertion (1)

$$A, B, \cdots ::= e_1 = e_2 \mid \Phi(e_1, \cdots, e_n) \mid A \bigwedge B \mid \neg A \mid \forall x.A$$

$$\begin{aligned} G ::= & \; p \longrightarrow p' : k(x : \alpha)\{A\}G \\ \mid & \; p \longrightarrow p' : k\{l_i\{A_i\} : G_i\} \\ \mid & \; (\mu t(\tilde{x})\{A\}.G)\langle \tilde{e} \rangle \\ \mid & \; t\langle \tilde{e} \rangle \\ \mid & \; \text{end} \end{aligned}$$

Global Assertion (1)

$$
\begin{aligned}
G_{neg} \;=\;& \texttt{rec}\, X\langle 100\rangle\,(prevp : \mathsf{int})\,\{prevp >= 100\}.\{ \\
& \quad \texttt{Buyer} \to \texttt{Seller}\colon k_1\,(p : \texttt{int})\{p >= prevp\}. \\
& \quad \texttt{Seller} \to \texttt{Buyer}\colon k_2\,\{ \\
& \qquad \mathbf{ok}\{p >= 200\}\colon G_{ok}, \\
& \qquad \mathbf{again}\{true\}\colon X\langle p\rangle \\
& \quad \}\} \\
G_{ok} \;=\;& \texttt{Buyer} \to \texttt{Bank}\colon k_3\,(v_p : \texttt{int})\{v_p == p\}. \\
& \quad \texttt{Bank} \to \texttt{Seller}\colon k_4\,(v_a : \texttt{bool})\{true\}.\mathsf{end}
\end{aligned}
$$

Consistency in Global Assertions (1)

- (History Sensitivity) *Carol cannot know* v.

$$\texttt{Alice} \rightarrow \texttt{Bob} : k_b(v : \texttt{int})\{true\}.$$

$$\texttt{Bob} \rightarrow \texttt{Carol} : k_c(w : \texttt{int})\{true\}.$$

$$\texttt{Carol} \rightarrow \texttt{Alice} : k_a(x : \texttt{int})\{x == v\}.\mathsf{end}$$

- (Temporal Satisfiability) *Bob can get stuck*.

$$\texttt{Alice} \rightarrow \texttt{Bob} : k_b(v : \texttt{int})\{v < 10\}.$$

$$\texttt{Bob} \rightarrow \texttt{Alice} : k_a(w : \texttt{int})\{v > w \;\&\&\; w > 6\}.\mathsf{end}$$

Consistency in Global Assertions (2)

Checking consistency (start from $C \equiv \mathsf{T}$):

- $p \longrightarrow p' : k(x : \alpha)\{A\}G$ is valid under C if C justifies $\exists x.A$ and G is valid under $C \wedge A$. Similarly for selection.
- $(\mu t(\tilde{x})\{A\}.G)\langle\tilde{e}\rangle$ is valid under C if C implies $A[\tilde{e}/\tilde{x}]$ and, moreover, G is valid under $A \wedge C$.
- $t\langle\tilde{e}\rangle$ with A invariant, is valid under C if C implies $A[\tilde{e}/\tilde{x}]$.

Consistency in Global Assertions (3)

Proposition.

The consistency of G is decidable if the underlying logic is decidable.

PROOF: For the side condition note we used implication in the underlying logic. ■

Endpoint Assertions

$$
\begin{array}{rcl}
T & ::= & k!(x:\alpha)\{A\}.T \\
& | & k?(x:\alpha)\{A\}.T \\
& | & k \oplus \{l_i\{A_i\} : T_i\} \\
& | & k\&\{l_i\{A_i\} : T_i\} \\
& | & (\mu t(\tilde{x})\{A\}.G)\langle \tilde{e} \rangle \\
& | & t\langle \tilde{e} \rangle \\
& | & \text{end}
\end{array}
$$

Projection (1)

$$
\begin{aligned}
G_{neg} \quad = \quad & \texttt{Buyer} \to \texttt{Seller}\colon k_1\,(v_o : \texttt{int})\{v_o >= 100\}. \\
& \texttt{Seller} \to \texttt{Buyer}\colon k_2\,\{ \\
& \qquad \mathbf{ok}\{v_o >= 200\}\colon G_{ok}, \\
& \qquad \mathbf{quit}\{\mathit{true}\}\colon \mathsf{end} \\
& \} \\
G_{ok} \quad = \quad & \texttt{Buyer} \to \texttt{Bank}\colon k_3\,(v_p : \texttt{int})\{v_o == v_p\}. \\
& \texttt{Bank} \to \texttt{Seller}\colon k_4\,(v_a : \texttt{bool})\{\mathit{true}\}. \\
& \mathsf{end}
\end{aligned}
$$

Projection (2)

$G_{neg} \upharpoonright \text{Buyer}$

$= \quad k_1!(v_o : \texttt{int})\{v_o >= 100\}.$

$k_2\&\{$

$\mathbf{ok}\{v_o >= 200\}: k_3!(v_p : \texttt{int})\{v_o == v_p\}.\mathsf{end},$

$\mathbf{quit}\{true\}: \mathsf{end}$

$\}$

$G_{neg} \upharpoonright \text{Bank}$

$= \quad k_3?(v_p : \texttt{int})\{\exists v_o.(v_o >= 200 \;\&\&\; v_o == v_p)\}.\mathsf{end}$

Consistency of Endpoint Assertions

Definition.

Consistency (temporal satisfiability) of endpoint assertions is defined similarly.

Proposition.

If G is consistent then each of its projection is consistent.

Proof System (1)

- *Sequent:*

$$\mathcal{C}; \Gamma \vdash P \triangleright \Delta$$

"Under a constraint $\mathcal{C}$ and a public contract Γ, the process P satisfies an endpoint session contract Δ."

- *Proof rules:* Elaborates the underlying typing rules. We can annotate processes along the way (here presented separately).

- *Semantics:* Through a conditional simulation.

Proof System (2): Session Initiation

$$\frac{\mathcal{C};\Gamma \vdash P \triangleright \Delta, \tilde{s}:(\Gamma(a) \upharpoonright \mathtt{p}) \mathbin{@} \mathtt{p} \quad \mathtt{p} \geq 1}{\mathcal{C};\Gamma \vdash a(\tilde{s})[\mathtt{p}].P \triangleright \Delta}$$

$$\frac{\mathcal{C};\Gamma \vdash P \triangleright \Delta, \tilde{s}:(\Gamma(a) \upharpoonright 1) \mathbin{@} 1 \quad n = \mathtt{maxrole}(G)}{\mathcal{C};\Gamma \vdash \overline{a}(\tilde{s})[2..n].P \triangleright \Delta}$$

Proof System (3): Linear Communications

$$\frac{\mathcal{C} \supset A[\tilde{e}/\tilde{x}] \quad \mathcal{C};\Gamma \vdash P \triangleright \Delta, \tilde{s}:T[\tilde{e}/\tilde{x}] \,@\, \mathtt{p} \quad \Gamma \vdash \tilde{e}:U}{\mathcal{C};\Gamma \vdash s_k!\langle \tilde{e} \rangle;P \triangleright \Delta, \tilde{s}:k!(\tilde{x}:\tilde{U})\{A\};T \,@\, \mathtt{p}}$$

$$\frac{\mathcal{C} \wedge A;\Gamma, \tilde{x}:U \vdash P \triangleright \Delta,\ \tilde{s}:T \,@\, \mathtt{p}}{\mathcal{C};\Gamma \vdash s_k?(\tilde{x}:U);P \triangleright \Delta,\ \tilde{s}:k?(\tilde{x}:U)\{A\};T \,@\, \mathtt{p}}$$

$$\frac{\mathcal{C} \supset A_j \quad \mathcal{C};\Gamma \vdash P \triangleright \Delta, \tilde{s}:T_j \,@\, \mathtt{p} \quad j \in I}{\mathcal{C};\Gamma \vdash s_k \lhd l_j : P \triangleright \Delta, \tilde{s}:k \oplus \{\{A_i\}l_i:T_i\}_{i \in I} \,@\, \mathtt{p}}$$

$$\frac{\mathcal{C} \wedge A_i;\Gamma \vdash P_i \triangleright \Delta, \tilde{s}:T_i \,@\, \mathtt{p} \quad \forall i \in I}{\mathcal{C};\Gamma \vdash\ s_k \rhd \{l_i:\ P_i\}_{i \in I} \triangleright \Delta, \tilde{s}:k\&\{\{A_i\}l_i:T_i\}_{i \in I} \,@\, \mathtt{p}}$$

Proof System (4): Recursion

$$\frac{}{\mathcal{C}\ ;\ \Gamma, X:(\tilde{v}:U)T\,@\,\mathrm{p} \vdash X\langle \tilde{e}\tilde{s}\rangle \triangleright \tilde{s}:T[\tilde{e}/\tilde{v}]\,@\,\mathrm{p}}$$

$$\frac{\mathcal{C}\ ;\ \Gamma, X:(\tilde{v}:U)T\,@\,\mathrm{p} \vdash P \triangleright \tilde{s}:T\,@\,\mathrm{p}}{\mathcal{C}\ ;\ \Gamma \vdash \mu X\langle \tilde{e}\tilde{s}\rangle(\tilde{v}\tilde{s}).P \triangleright\ \tilde{s}:T[\tilde{e}/\tilde{v}]\,@\,\mathrm{p}}$$

Proof System (5): Consequence

$$\frac{\mathcal{C}';\Gamma \vdash P \triangleright \Delta' \quad \mathcal{C} \supset \mathcal{C}' \quad \Delta' \ni \Delta}{\mathcal{C};\Gamma \vdash P \triangleright \Delta}$$

Refinement (1)

- "T_a refines T_b" means T_a is a stronger (more constraining, more detailed) specification than T_b.
- If P satisfies T_a (at s) and T_a refines T_b, then P should also satisfy a weaker specification T_b.
- The notion of "stronger" specifications comes from two sources: *subtyping* and *interaction predicates*.

Refinement (2)

Record subtyping. We know a class (record) with signature

$$[l_1 : \alpha_1,\ l_2 : \alpha_2,\ l_3 : \alpha_3]$$

is a subtype of (or refines)

$$[l_1 : \alpha_1,\ l_3 : \alpha_3]$$

Why? Because the former can receive more invocations (is more generous, more gentle) than the latter.

Refinement (3)

- In the same way

$$\&[l_1 : T_1,\ l_2 : T_2,\ l_3 : T_3]$$

is a subtype of

$$\&[l_1 : T_1,\ l_3 : T_3]$$

- Dually

$$\oplus[l_1 : T_1,\ l_3 : T_3]$$

is a subtype of

$$\oplus[l_1 : T_1,\ l_2 : T_2,\ l_3 : T_3].$$

Refinement (4)

- Suppose we have two output types, $k!\texttt{Real}.T$ and $k!\texttt{Int}.T$. Which is more gentle?
- Dually for $k?\texttt{Real}.T$ and $k?\texttt{Int}.T$.
- In $\alpha.T$, T is naturally covariant.

Refinement (5)

- Finally we consider a predicate-level refinement.
  Given $k!(x{:}\texttt{int})\{x = 2\}.T$ and $k!(x{:}\texttt{int})\{Even(x)\}.T$,
  which specifies a more gentle behaviour?
- Given $k?(x{:}\texttt{int})\{Odd(x)\}.T$ and $k?(x{:}\texttt{int})\{x = 1\}.T$,
  which refines which? Which freaks out more often?
- Similarly for selection ($\oplus$) and branching (&).

Refinement (6)

Formally: $T_a \ni T_b$ if $(T_a, T_b) \in \mathcal{R}$ where whenever $\mathcal{T}_1 \mathcal{R} \mathcal{T}_2$, we have:

- $T_1 = k!(\tilde{v} : T \,@\, \mathrm{p})\{A_1\}; T_1'$, $T_2 = k!(\tilde{v} : T' \,@\, \mathrm{p})\{A_2\}; T_2'$ s.t. $A_1 \supset A_2$, $T_1' \sigma \mathcal{R}\, T_2' \sigma$ and $T \mathcal{R}\, T'$ for each $\sigma \models A_1$.
- $T_1 = k?(\tilde{v} : T \,@\, \mathrm{p})\{A_1\}; T_1'$ and $T_2 = k?(\tilde{v} : T' \,@\, \mathrm{p})\{A_2\}; T_2'$ s.t. $A_2 \supset A_1$, $T_1' \sigma \mathcal{R}\, T_2' \sigma$ and $T' \mathcal{R}\, T$ for each $\sigma \models A_2$.
- $T_1 = k \oplus \{\{A_{1i}\} l_{1i} : T_{1i}\}_{i \in I}$ and $T_2 = k \oplus \{\{A_{2j}\} l_{2j} : T_{2j}\}_{j \in J}$ where $I \subset J$, $A_{1i} \supset A_{2i}$ and $T_{1i} \mathcal{R}\, T_{2i}$ $(i \in I)$.
- $T_1 = k \& \{\{A_{1i}\} l_{1i} : T_{1i}\}_{i \in I}$ and $T_2 = k \& \{\{A_{2j}\} l_{2j} : T_{2j}\}_{j \in J}$ where $J \subset I$, $A_{2j} \supset A_{1j}$ and $T_{1j} \mathcal{R}\, T_{2j}$ $(j \in J)$.

Asserted Processes

We annotate typed processes to obtain asserted processes, i.e. processes whose actions are checked at runtime.

$$\frac{\Gamma \vdash P \triangleright \Delta, \tilde{s} : T[\tilde{e}/\tilde{v}] \mathbin{@} \mathtt{p}}{\Gamma \vdash s_k!\langle \tilde{e} \rangle (\tilde{v} : U)\{A\}; P \triangleright \Delta, \tilde{s} : k!(\tilde{v})\{A\}; T \mathbin{@} \mathtt{p}}$$

$$\frac{\Gamma, \tilde{v} : U \vdash P \triangleright \Delta,\ \tilde{s} : T \mathbin{@} \mathtt{p}}{\Gamma \vdash s_k?(\tilde{v} : U)\{A\}; P \triangleright \Delta,\ \tilde{s} : k?(\tilde{v} : U)\{A\}; T \mathbin{@} \mathtt{p}}$$

An annotated process can have an assertion error, e.g.:

$$s!\langle 3 \rangle (x : \mathtt{int})\{Even(x)\}; P \,|\, s : \tilde{h} \longrightarrow \mathsf{err} \,|\, s : \tilde{h}$$

Labelled Transition

LTS: “Behaviour of processes w.r.t. queues”

$$l ::= a[i](\tilde{s}) \mid \overline{a}[2..n](\tilde{s}) \mid s!v \mid s?v \mid (\nu\, a)s!\langle a\rangle \mid s \triangleleft l \mid s \triangleright l \mid \tau$$

$$s?(x).P \xrightarrow{s?v} P[v/x]$$

$$s!v;P \xrightarrow{s!v} P$$

$$s \triangleright [l_i : P_i] \xrightarrow{s \triangleright l_j} P_j$$

$$s \triangleleft l;P \xrightarrow{s \triangleleft l} P$$

Semantics of Judgement

A binary relation $\mathcal{R}$ relating a closed process P and a pair $\langle \Gamma, \Delta \rangle$, assuming well-typedness, is a *conditional simulation* if, for each $(P, \langle \Gamma, \Delta \rangle) \in \mathcal{R}$:

1. for each input/branching/session input $P \xrightarrow{\alpha} P'$, we have if $\langle \Gamma, \Delta \rangle \xrightarrow{\alpha} \langle \Gamma', \Delta' \rangle$ then $(P', \langle \Gamma', \Delta' \rangle) \in \mathcal{R}$.
2. for each output/selection/τ/session output $P \xrightarrow{\alpha} P'$, we have $\langle \Gamma, \Delta \rangle \xrightarrow{\alpha} \langle \Gamma', \Delta' \rangle$ such that $(P', \langle \Gamma', \Delta' \rangle) \in \mathcal{R}$.

We write $\Gamma \models P \triangleright \Delta$ if $(P, \langle \Gamma, \Delta \rangle) \in \mathcal{R}$ for some conditional simulation $\mathcal{R}$.

Main Results (1) Soundness

Theorem. (soundness)

If $\Gamma \vdash P \triangleright \Delta$ *for a closed process* P*, then* $\Gamma \models P \triangleright \Delta$.

PROOF: By rule induction. ■

Corollary. (error-freedom)

If $\Gamma \vdash P \triangleright \Delta$ *for a closed process* P*, then* P *does not cause an assertion error after any number of reductions*.

Main Results (2) Completeness

Definition. (visibility)

A process is visible if its hidden name is immediately exported by bound output.

Theorem. (completeness)

If P *is visible and* $\Gamma \models P \triangleright \Delta$ *for a closed process* P*, then* $\Gamma \vdash P \triangleright \Delta$.

PROOF: By constructing the rules for generating "principal assertions" for visible processes which are instances of the proof rules: by Consequence rule, we obtain the required assertion, giving a decidable algorithm for $\models$ for visible processes as far as the underlying logic is decidable. ■

Main Results (3) Effective Validation

Definition. (restricted refinement)
A *restricted refinement relation* is defined as before restricting comparison between recursive assertions to those with the same shape.

Proposition. (effective validation)
Under the same condition and using the restricted refinement relation in the Consequence Rule, the relation $\Gamma \vdash P \triangleright \Delta$ *is decidable if the underlying logic is decidable.*

PROOF: Through the construction of principal assertions and noting that the restricted refinement is decidable if the underlying logic is decidable. ■

Discussions (1)

- Corresponding assertions [BCG04]: Integration of session types and corresponding assertions.
- Many other related work in web-service contracts cf. [Castagna, Laneve, Padovani, ..].
- The framework easily extends to delegation (session channel passing).
- Refined assertions may be suggested by more expressive behavioural logic, such as HML (into which the semantics and proof rules can be embedded).
- DbC for π can embed the traditional DbC.

Embedding Traditional DbC

```
int foobar(str s) {
   pre: { s.length()<=100 }
   post:{ result <=1000 }
}
```

becomes

```
protocol foobar {
  participants Server, Client;
  s:str {s.length()<=100} from Client to Server;
  res:int {res<=1000} from Server to Client;
}
```

Discussions (2)

- Effective runtime checking is a further topic.
- Treatment of recursive assertions is a key to many engineering challenges.
- Applications are being considered centring on Scribble (which is based on [BCDDDY08]).