Inference of progress properties for (multi party) sessions

Mario Coppo (Universita’ di Torino)

joint work with

Mariangiola Dezani, Nobuko Yoshida

Lisbon, April 19, 2011

Aim of the talk

In a previous paper on multiparty session types

[1] *Bettini, Coppo, M, D'Antoni, Dezani-Ciancaglini, Yoshida, Global progress in dynamically interleaved multiparty sessions. (CONCUR 2008)*

We introduced a type system to assure global progress for interleaved multiparty sessions.

The “interaction” type system of [1] has a non deterministic rules (i.e. multiple choices for the same linguistic construct) and so type inference is not easy (unless to resort to backtracking)

In this talk we will discuss some principles for the design of a deterministic syntax directed inference procedure for checking typability in the interaction type system.

The inference procedure can also be considered as a stand-alone analysis tool.

Our starting point

To focus on the relevant ideas in this talk we will do some simplifying assumptions:

• we will consider only binary session (generalization to multiparty session is rather easy)

• we will consider a simplified language, including only session opening and communications (no branches and conditional, no recursion, no delegation).

A basic assumption: we consider only terms *typable* in the *standard communication type system*.

(note that typability in the communication type system guarantees progress of each single session in insulation).

Progress property

Let P be a closed process (typable in the communication type system). Then P has the *progress property* if P $\rightarrow^*$ P' implies that

- either P' = **0**
- or P' | Q $\rightarrow$ R for some Q such that P | Q is well typed in the communication type system, Q $\not\rightarrow$ and R has in its turn the progress property

(Roughly speaking: P has the progress property if it cannot reduce to a process hopelessly blocked in a deadlock.)

A “minimal” session language

Language (basic)

P	:=	0	null
		c(x).P	session request
		$\overline{c}$(x).P	session accept
		s!<e>.P	data output
		s?<x>.P	data input
		(ν a). P	shared channel hiding
		(ν s). P	live channel hiding
		P | Q	parallel composition

Session opening rule:

$$a(x).P \mid \overline{a}(y).Q \rightarrow (\nu\ s).\ P[s/x] \mid Q[s/y]$$

Communication is **asyncronous**.

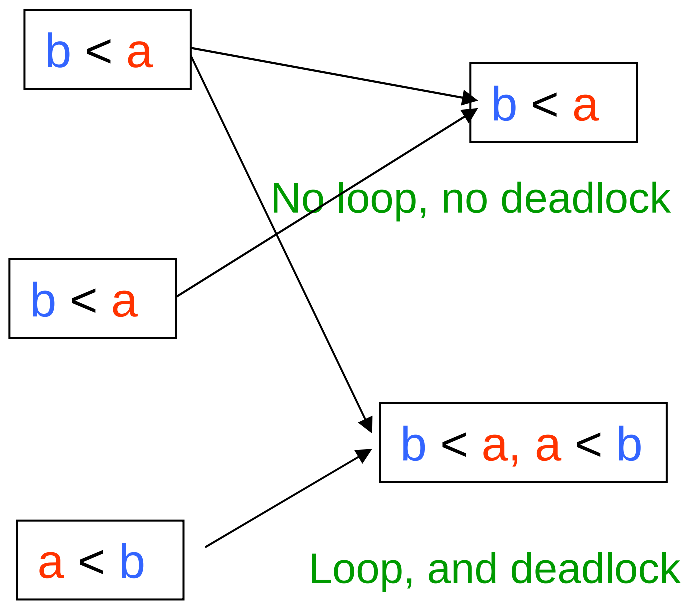
Main idea: keep a relation **R** on channels (and service) names with the following meaning:

- x < y   an input action on *channel* x precedes an in-out action on y
- a < y   an input action of *service* a precedes an in-out action on channel y
- similarly y < a , a < b

P : b(y). a(x). y?(w). x!<2>. y!<5>. x?(t) .0

Q1 : $\overline{b}$(y). $\overline{a}$(x). x!<4>. y!<true>. y?(t). x?(w). 0

Q2 : $\overline{b}$(y). $\overline{a}$(x). x?(w). x!<2>. y?(t). y!<true>



Building the R relation

The order relation between cannels and sessions is built by analyzing the process structural way.

y?(w). x!<2>. y!<5>. x?(t) 0

R:

$$\begin{tikzcd} x \arrow[d] \\ y \end{tikzcd}$$

Then $y < x$

$\dashrightarrow$ means >

Building the R relation

The order relation between cannels and sessions is built by analyzing the process structural way.

b[y]. a[x]. y?(w). x!<2>. y!<5>. x?(t) 0

R:

$$\begin{tikzcd} a \arrow[d] \\ b \end{tikzcd}$$

Then b < a

$\dashrightarrow$ means >

Nested sessions never deadlock

Note: when both participants have a nested structure deadlock is not possible:

P: a(x). x?(t). b(y). y!<2>. y?(u). x?(w). 0

Q : $\overline{a}$(x). x!<3>. $\overline{b}$(y). y?(u). y!<4>. x!<5>. 0

$$\begin{tikzcd}
\boxed{a < b < a} \arrow[dr] & \\
& \boxed{a < b < a} \\
\boxed{b < a} \arrow[ur] &
\end{tikzcd}$$

Sessions are nested in both processes

There is a loop but no deadlock

A general principle

the deadlock freedom of nested session is a particular case of a more general principle: if, when closing a session, its name is minimal in the R relation, it can be safely *eliminated* from the order.

Warning (see later): provided that this is done in both participants!

Using this principle no loop could be detected in nested sessions:

b[y]. y!<2>. y?(u). x?(w). 0

R:

$$\begin{tikzcd} x \arrow[d] \\ b \end{tikzcd}$$

But b is minimal, so we can erase it from **R**

A general principle

the deadlock freedom of nested session is a particular case of a more general principle: if, when closing a session, its name is minimal in the R relation, it can be safely *eliminated* from the order.

Warning (see later): provided that this is done in both participants!

Using this principle no loop could be detected in nested sessions:

a[x]. x?(t). b[y]. y!<2>. y?(u). x?(w). 0

R:

Also a would be minimal, so in the end the relation **R** results empty

We want so to consider these basic principles

- loop freedom guarantees progress
- service names can be removed (with restrictions…) when “minimal” in the order

The analysis is performed in a structural way (like usual type checking) so the inference rules follows the process formation bottom-up. The basic strategy is the following:

-Take trace of the precedence relation as the analysis go on and until no loop is detected;

- As soon as a loop is detected see if there is (minimal subset) of sessions that are indeed nested and see if removing them we can get free of the loop.

A too simple strategy ….

Assume we are analyzing the process:

c[z]. a[x]. x?(v). b[y]. y?(w). z!<1>. y!<2>. x!<3>. 0

After analyzing: x?(v). b[y]. y?(w). z!<1>. y!<2>. x!<3>. 0

we get:

R:

$$\begin{tikzcd}
x \arrow[dr, shift left] & & z \arrow[dl] \\
& b \arrow[ul, shift left] &
\end{tikzcd}$$

We are getting in a loop…….but we can refine our strategy

$\longrightarrow$ means $>$

At the moment of processing b[y] we see that b would be *minimal* in the order, so we could remove it from the order. *But* this will commit us to handle in the same way also the service $\overline{b}$.

We use a *dependency relation* C *to remember that b was minimal at the moment of its introduction*.

x?(v). b[y]. y?(w). z!<1>. y!<2>. x!<3>. 0

Loop!

R:

$$\begin{tikzcd} x & z \arrow[l] \end{tikzcd}$$

C : {(b, {}),}

N = { b }

In order to remember that b has been erased from **R** we put b in set N (the “nesting” set) . We remove also b from C

Going on ...

c[z]. a[x]. x?(v).b[y]. y?(w). z!<1>. y!<2>. x!<3>. 0

R:

$$\begin{tikzcd}
c \arrow[d] \\
a
\end{tikzcd}$$

C = { (a, { }), (c, {a})}}

N = { b }

Now closing c we observe that:

Again a is now minimal. So we remember this in C

- we have no loops and

- c *could* be minimal if a where handled with the minimality rule

We register this using the relation C.

In the and **R**, C, N are the result of the analysis of this process

Parallel composition I

P1 : a[x]. x!<3> b[y]. y?(t). y!<2>. x?(v). 0

$\mathbf{R_{P1}}$:

$$\begin{tikzcd} a \arrow[d] \\ b \end{tikzcd}$$

C_1 = { (b, {}), (a, {b}) }

N_1 = {}

P2 : $\overline{a}$[x]. x?(u). $\overline{b}$[y]. y!<2>. y?(u). x!<1>. 0

$\mathbf{R_{P2}}$:

C_2 = { }

N_2 = {a, b}

For the parallel composition P1 | P2 we must impose that to a and b is applied the minimality principles moving them to N

$\mathbf{R}_{P1|P2}$:

C1 = { }

N1 = {a, b}

Note: this will be particularly important in the case of multiparty sessions

Parallel composition II

P1 : a[x]. x?(t). b[y]. x!<1>. y?(t). 0

$\mathbf{R_{P1}}$:

$$\boxed{\begin{tikzcd} b \arrow[d] \\ a \end{tikzcd}}$$

$\mathbf{C_1}$ = { (b, { }), (a, { }) }

N_1 = { }

P2 : $\overline{a}$[x]. x!<2>. $\overline{b}$[y]. x?(u). y!<2>. 0

$\mathbf{R_{P2}}$:

$$\boxed{\begin{tikzcd} b \arrow[d] \\ a \end{tikzcd}}$$

$\mathbf{C_2}$ = { (a, { }) }

N_2 = { }

For the parallel composition P1 | P2 we must exclude the possibility of applying the minimality principles moving b to N

P1 | P2

$\mathbf{R}_{P1|P2}$:

$$\boxed{\begin{tikzcd} b \arrow[d] \\ a \end{tikzcd}}$$

C = { (a, { }) }

N = { }

Going on in the analysys we will have the possibility of moving in N only a.

On the need of C and N

Remark: it is crucial that we keep the sets C, N in order to control the way in which minimal names are eliminated from **R.** In the following processes P1, P2 we use the minimality condition independently in P1 but not in P2. .

P1 : a[x]. b[y]. y?(t). y!<2>. x!<3> x?(v). 0   **R_{P1}:** $\boxed{}$

both b and a are minimal when closed

b and a are not nested and we cannot eliminate b from **R**

P2 : $\overline{a}$[x]. $\overline{b}$[y]. x?(v). x!<3>. y!<2>. y?(t). 0   **R_{P1}:** $\boxed{b}$

Putting P1 and P2 in parallel we have no loop but P1 | P2 get stuck.

Using C and N we get the inconsistency

P1 : a[x]. b[y]. y?(t). y!<2>. x!<3> x?(v). 0

R_{P1}: $\boxed{b < a}$

CP1:{((b, { }), (a, {b}) }

NP1 = { }

P2 : $\overline{a}$[x]. $\overline{b}$[y]. x?(v). x!<3>. y!<2>. y?(t). 0

R_{P2}: $\boxed{a < b}$

CP2:{(a, { }) }

NP2 = { }

To eliminate the inconsistency we should close a and b in *both* processes with the minimal rule, but we cannot do that for b in P2. So we get b < a < b

In the full system

The inference procedure there are more aspects and language extensions that are considered in the paper. The most important are:

• *delegation* (some restrictions are needed for delegated sessions)

• possibility of having *communication and restrictions of service names* (a new set B will be added for this).

• *internal and external choices* and if-then-else.

• *multiparty* sessions are considered and not binary ones (not really different ideas but generalizations of the considered strategy).

• *recursion*

Some inference rule

Judgment: $\Theta\,;P \Mapsto \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}$

$$\frac{\Theta\,;P \Mapsto \mathsf{R}\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B} \qquad \mathsf{R}\downarrow y \subseteq dom(\mathsf{C})}{\Theta\,;y?(\mathsf{q},x);P \Mapsto \mathsf{pre}(y,\mathsf{R}\setminus(\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y)))\,;\,\mathsf{C}\setminus(\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y));\,\mathsf{N}\cup(\mathsf{rcl}(\mathsf{C},\mathsf{R}\downarrow y));\,\mathsf{B}}\ \{\textsc{Rcv-I}\}$$

$$\frac{\Theta\,;P_i \Mapsto \mathsf{R}_i\,;\,\mathsf{C}_i\,;\,\mathsf{N}_i\,;\,\mathsf{B}_i \quad i\in I \quad \mathsf{R};\mathsf{C};\mathsf{N};\mathsf{B} = h(\{\mathsf{R}_i;\mathsf{C}_i;\mathsf{N}_i;\mathsf{B}_i \mid i\in I\})}{\Theta\,;y\&(\mathsf{p},\{l_i:P_i\}_{i\in I}) \Mapsto \mathsf{pre}(y,\mathsf{R})\,;\,\mathsf{C};\,\mathsf{N};\,\mathsf{B}}\ \{\textsc{Branch-I}\}$$

where $h(\{\mathsf{R}_i;\mathsf{C}_i;\mathsf{N}_i;\mathsf{B}_i \mid i\in I\})$ = (R; C; N; B) and

$$\mathsf{R} = (\textstyle\bigcup_{i\in I}\mathsf{R}_i)\setminus\mathsf{N} \qquad \mathsf{C} = (\textstyle\Cup_{i\in I}\mathsf{C}_i)\setminus(\mathsf{N}\cup\mathsf{K}) \qquad \mathsf{N} = \textstyle\bigcup_{i\in I}\mathsf{N}_i\cup\mathsf{cl}(\Cup_{i\in I}\mathsf{C}_i,\bigcup_{i\in I}\mathsf{N}_i)$$

$$\mathsf{B} = \{a \mid \exists i\in I.\ a\in\mathsf{B}_i\ \wedge\ \forall i\in I.\ (a\in\mathsf{B}_i \vee a\notin dom(\mathsf{R}_i)\cup\mathsf{N}_i)\}$$

if $\mathsf{N}\cap\mathsf{K}=\emptyset$ where

$$\mathsf{D}_i = dom(\mathsf{R}_i)\setminus dom(\mathsf{C}_i)\ \ i\in I \qquad \mathsf{K} = \mathsf{rcl}^{\Uparrow}(\Cup_{i\in I}\mathsf{C}_i,\bigcup_{i\in I}\mathsf{D}_i)$$

The overall proof strategy

We have first designed a (non deterministic) interaction type with judgments:

$$\Theta \vdash P \;\blacktriangleright\; \mathscr{R} \,;\, \mathscr{N} \,;\, \mathscr{B}$$

proving that a process has the progress property iff it is typable in the interaction type system.

Then we have proved that the deterministic inference procedure outlined in these slides is sound and complete for the interaction type system: i.e. a process is typable by the inference rules iff it has the progress property.

Further steps

- Integrate progress type system with the communication type system.
- Adapt the inference procedure to recent extensions of session types (dynamic multirole [Danielou, Yoshida] , ….)
- Simplify the inference procedure……?