

# Towards Trustworthy Multiparty Sessions

Roberto Bruni<sup>1</sup>   Ivan Lanese<sup>2</sup>   Hernán Melgratti<sup>3</sup>  
Leonardo Gaetano Mezzina<sup>4</sup>   Emilio Tuosto<sup>5</sup>

<sup>1</sup>Università di Pisa

<sup>2</sup>Università di Bologna

<sup>3</sup>Universidad de Buenos Aires

<sup>4</sup>IMT Alti Studi Lucca

<sup>5</sup>University of Leicester

PLACES 2008  
Oslo, Norway  
June 7th, 2008

- 1 Introduction & Motivation
- 2 A Glimpse of  $\mu se$ , graphically
- 3 Something About Types
- 4 Concluding Remarks

# A Shared Vision (Hopefully!)

## 1st Fact

*Trustworthy Service Oriented Computing is hard:* services are autonomous, heterogeneous, separately designed computational entities to be dynamically assembled.

## 2nd Fact

*Process Calculi can help:* they allow to focus on salient features at a convenient level of abstraction.

## 3rd Fact

*Behavioural types can help:* syntactic descriptions of services are not expressive enough to guarantee their trustworthy assembly.

## 4th Fact (or mere conjecture?)

*Existing techniques must be adapted:* SOC has specific features like endpoints, sessions, dynamicity.

# A Shared Vision (Hopefully!)

## 1st Fact

*Trustworthy Service Oriented Computing is hard:* services are autonomous, heterogeneous, separately designed computational entities to be dynamically assembled.

## 2nd Fact

*Process Calculi can help:* they allow to focus on salient features at a convenient level of abstraction.

## 3rd Fact

*Behavioural types can help:* syntactic descriptions of services are not expressive enough to guarantee their trustworthy assembly.

## 4th Fact (or mere conjecture?)

*Existing techniques must be adapted:* SOC has specific features like endpoints, sessions, dynamicity.

# A Shared Vision (Hopefully!)

## 1st Fact

*Trustworthy Service Oriented Computing is hard:* services are autonomous, heterogeneous, separately designed computational entities to be dynamically assembled.

## 2nd Fact

*Process Calculi can help:* they allow to focus on salient features at a convenient level of abstraction.

## 3rd Fact

*Behavioural types can help:* syntactic descriptions of services are not expressive enough to guarantee their trustworthy assembly.

## 4th Fact (or mere conjecture?)

*Existing techniques must be adapted:* SOC has specific features like endpoints, sessions, dynamicity.

# A Shared Vision (Hopefully!)

## 1st Fact

*Trustworthy Service Oriented Computing is hard:* services are autonomous, heterogeneous, separately designed computational entities to be dynamically assembled.

## 2nd Fact

*Process Calculi can help:* they allow to focus on salient features at a convenient level of abstraction.

## 3rd Fact

*Behavioural types can help:* syntactic descriptions of services are not expressive enough to guarantee their trustworthy assembly.

## 4th Fact (or mere conjecture?)

*Existing techniques must be adapted:* SOC has specific features like endpoints, sessions, dynamicity.

# Our Proposal

## $\mu$ se (after MULTiparty SEssions)

$\mu$ se is a process calculus for expressing computations where endpoints dynamically join existing multiparty sessions (as seen on Emilio's talk @ COORDINATION 2008)

## Types for

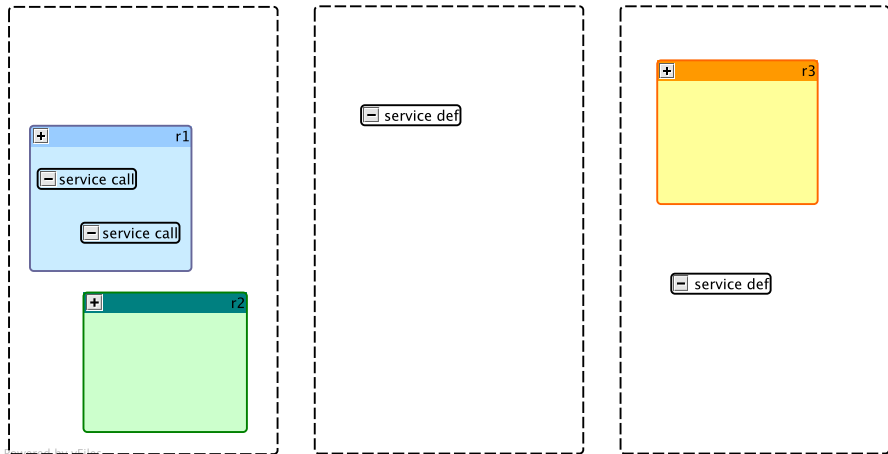
- Semantic description of services (for discovery)
- Compatibility check (for dynamic assembly)
- Early detection of possible sources of problems (trustworthiness)

## Disclaim

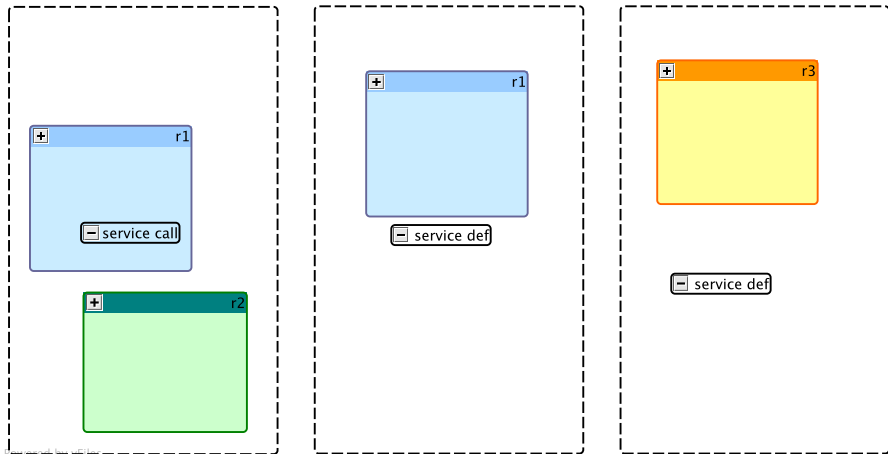
- We restrict to consider a “bare bones” fragment of  $\mu$ se
- We present a parametric type system w.r.t. 3 notions (task separation, dual type compatibility, session completion)
- We conjecture subject reduction + all non-typeable processes can deadlock
- We look for stronger guarantees

- 1 Introduction & Motivation
- 2 A Glimpse of  $\mu se$ , graphically
- 3 Something About Types
- 4 Concluding Remarks

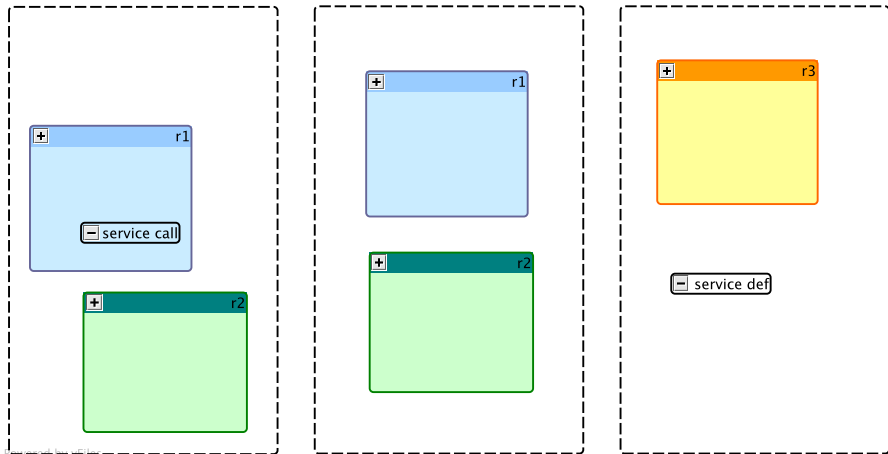
# Sites, Service Invocation and Sessions



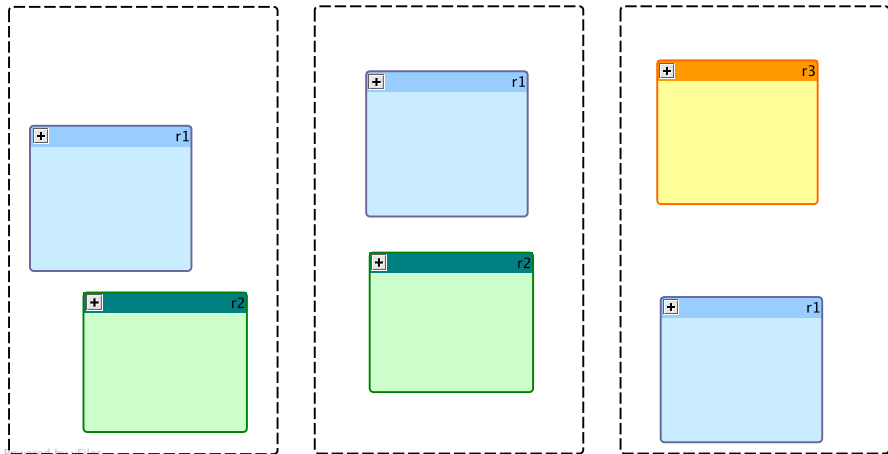
# Sites, Service Invocation and Sessions



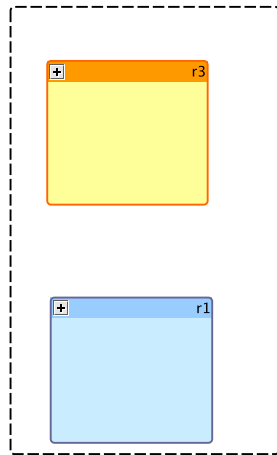
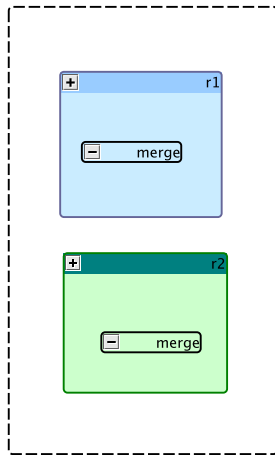
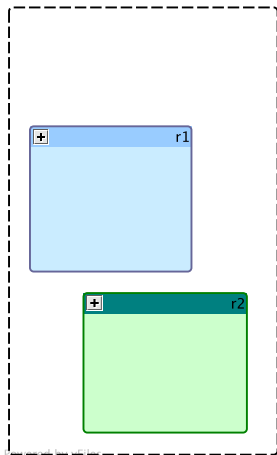
# Sites, Service Invocation and Sessions



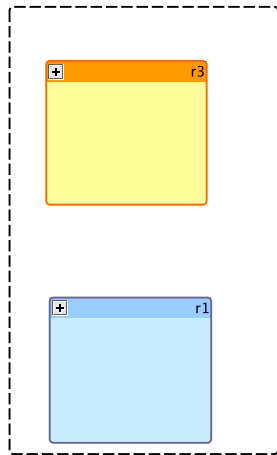
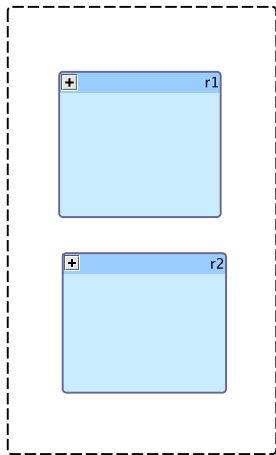
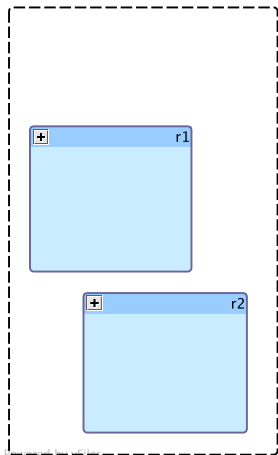
# Sites, Service Invocation and Sessions



# Merging Sessions



# Merging Sessions



- 1 Introduction & Motivation
- 2 A Glimpse of  $\mu se$ , graphically
- 3 Something About Types**
- 4 Concluding Remarks

# Types for Dynamic Multiparty Sessions

## Type judgments

$$\Gamma; \Delta \vdash P : \{\sigma \nearrow \rho\}$$

- we call  $\sigma$  the *current type*, and  $\rho$  the *delegated type*
- $P$  provides communication activities in  $\sigma$  and  $\rho$
- activities  $\sigma$  concern the current participants of its session,
- activities  $\rho$  concern other endpoints that  $P$  itself will allow to join its session (via service invocation or merge)
- $\Gamma$  is a finite partial mapping from variables  $X$  and polarized service / entry-point names  $n^p$  (with  $p \in \{+, -\}$ ) to type pairs  $\sigma \nearrow \rho$ , with the understanding that actions in  $\rho$  are delegated to  $n^{\bar{p}}$ .
- $\Delta$  is a finite partial mapping from session names  $r$  to types  $\sigma$ , such that  $\Delta(r)$  is the parallel composition of the current types of all endpoints of  $r$

## Self typeable systems

$$\Gamma; \Delta \vdash S : \{0 \nearrow 0\}$$

# Parametric on Task Separation

## Task separation

Task separation  $c * \sigma$  is used to project the activities of  $P$  in separate threads for later delegation.

## Our choice

Here we take the most relaxed form of separation, where  $c * \sigma = c | \sigma$ .

## Used in

$$\begin{array}{c} \text{(T}_{\text{ACTION}}\text{)} \\ \Gamma; \Delta \vdash P : \{\sigma \nearrow \rho\} \\ \hline \Gamma; \Delta \vdash c.P : \{c * \sigma \nearrow \rho\} \end{array}$$

# Parametric on Type Compatibility

## Type compatibility

Type compatibility  $\sigma \approx \rho$  says that  $\sigma$  and  $\rho$  are complementary.

## Our choice

Let  $I(\sigma) = \{c \mid \exists \sigma' : \sigma \xrightarrow{c} \sigma'\}$  be the set of initial actions of  $\sigma$ .

Here we take the largest relation on types such that whenever  $\sigma \approx \rho$ :

- either  $I(\sigma) = I(\rho) = \emptyset$ ,
- or  $K = I(\sigma) \cap \overline{I(\rho)} \neq \emptyset$  and, for each  $x \in K$  and for each  $\sigma'$  and  $\rho'$  such that  $\sigma \xrightarrow{x} \sigma'$  and  $\rho \xrightarrow{\bar{x}} \rho'$ , then  $\sigma' \approx \rho'$ .

## Used in

$\Gamma$  is *well-formed* if:

- whenever  $\Gamma(n^p) = \sigma \nearrow \rho$ , then  $\Gamma(n^{\bar{p}}) = \sigma' \nearrow \rho'$  for some  $\rho' \approx \rho$ ,
- whenever  $\Gamma(a^-) = \sigma \nearrow \rho$ , then  $\sigma = 0$ .

# Parametric on Session Completion

## Session completion

The completion set  $\Downarrow_0$  contains those types  $\sigma$  that express admissible interactions of multiple endpoints.

## Our choice

Here we define  $\Downarrow_0$  as the largest set of types  $\sigma$  such that:

- for each  $c \in I(\sigma)$  such that  $\bar{c} \notin I(\sigma)$  and for each  $\sigma \xrightarrow{\tau} \sigma'$  there exists  $\sigma''$  such that  $\bar{c} \in I(\sigma'')$  and  $\sigma' \xrightarrow{\tau^*} \sigma''$ ,
- if  $\sigma \xrightarrow{\tau} \sigma'$  then  $\sigma' \in \Downarrow_0$ .

## Used in

We say that  $\Delta$  is *fully-formed* if whenever  $\Delta(r) = \sigma$ , then  $\sigma \in \Downarrow_0$ .

## Example: Two Buyers

$(\nu r_1, r_2)(l_s :: Sell \mid l_1 :: Buy_1 \mid l_2 :: Buy_2)$

$Sell = sell \Rightarrow title.install[Offer].merge^- e$

The service *sell* waits for a buyer to require a quote for a book (*title*), installs a new service *offer* for a second buyer and prepares for merging with an instance of *offer*.

$Offer = offer \Rightarrow merge^+ e.(\overline{title}.(\overline{quote}.Q_3|\overline{quote}.Q_4))$

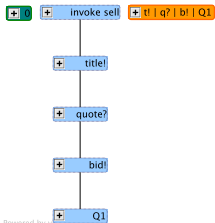
*offer* provides the book's *title* so that quotes are communicated to both buyers after the sessions are merged.

$Buy_1 = r_1 \triangleright invoke\ sell.\overline{title}.quote.\overline{bid}.Q_1$

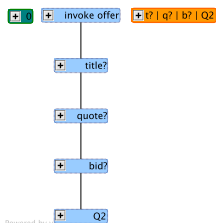
$Buy_2 = r_2 \triangleright invoke\ offer.\overline{title}.quote.\overline{bid}.Q_2$

Buyers communicate over *bid* and the negotiation is concluded by the interactions among *Q*, *Q*<sub>1</sub> and *Q*<sub>2</sub> (not modeled here).

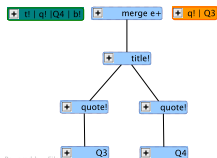
# Example: Typing the Two Buyers



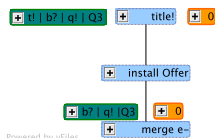
Powered by yFiles



Powered by yFiles



Powered by yFiles



Powered by yFiles

$$\Gamma = \left\{ \begin{array}{ll} \text{sell}^+ : (0 \nearrow \text{b} \mid \text{t} \mid \bar{\text{q}} \mid \text{Q}_3), & \text{sell}^- : (0 \nearrow \bar{\text{b}} \mid \bar{\text{t}} \cdot \text{q} \mid \text{Q}_1), \\ \text{offer}^+ : (0 \nearrow \bar{\text{b}} \mid \bar{\text{t}} \mid \bar{\text{q}} \mid \text{Q}_4), & \text{offer}^- : (0 \nearrow \text{b} \mid \text{t} \mid \text{q} \mid \text{Q}_2), \\ \text{e}^- : (\bar{\text{b}} \nearrow 0), & \text{e}^+ : (\bar{\text{q}} \mid \text{b} \mid \text{Q}_3 \nearrow 0) \end{array} \right\}$$

(TINSTALL)

$$\frac{\Gamma; \Delta \vdash P : \{\sigma_1 \mid \sigma_2 \nearrow \rho\} \quad \Gamma(a^+) = \sigma_1 \nearrow \sigma_2 \quad \Gamma(a^-) = 0 \nearrow \rho' \quad \Gamma; \Delta \vdash Q : \{\Phi\}}{\Gamma; \Delta \vdash \text{install}[a \Rightarrow P].Q : \{\Phi\}}$$

(TMERGE)

$$\frac{\Gamma; \Delta \vdash P : \{\sigma_1 \mid \sigma_2 \mid \sigma_3 \nearrow \rho\} \quad \Gamma(e^P) = \sigma \mid \sigma_2 \nearrow \sigma_3 \quad \Gamma(e^{\bar{P}}) = \sigma' \mid \sigma'' \nearrow \rho' \quad \sigma \approx \sigma''}{\Gamma; \Delta \vdash \text{merge}^P e.P : \{\sigma' \mid \sigma_1 \mid \sigma'' \nearrow \sigma_2 \mid \sigma_3 \mid \rho\}}$$

- 1 Introduction & Motivation
- 2 A Glimpse of  $\mu se$ , graphically
- 3 Something About Types
- 4 Concluding Remarks

# Conclusion

## Preliminary study

Some self typeable systems can behave “badly” (no check on availability of services, entry points, etc).

## Future work

- Suitable syntactic restrictions to obtain stronger guarantees.
- Change the notions of  $c * \sigma$ ,  $\approx$  and  $\Downarrow_0$  if needed

Your feedback would be welcome!!

## Preliminary study

Some self typeable systems can behave “badly” (no check on availability of services, entry points, etc).

## Future work

- Suitable syntactic restrictions to obtain stronger guarantees.
- Change the notions of  $c * \sigma$ ,  $\approx$  and  $\Downarrow_0$  if needed

**Your feedback would be welcome!!**