Towards Trustworthy Multiparty Sessions

Roberto Bruni[1] Ivan Lanese[2] Hernán Melgratti[3]
Leonardo Gaetano Mezzina[4] Emilio Tuosto[5]

[1]Università di Pisa

[2]Università di Bologna

[3]Universidad de Buenos Aires

[4]IMT Alti Studi Lucca

[5]University of Leicester

PLACES 2008
Oslo, Norway
June 7th, 2008

Outline

A Shared Vision (Hopefully!)

1st Fact

Trustworthy Service Oriented Computing is hard: services are autonomous, heterogeeneous, 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, heterogeeneous, 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, heterogeeneous, 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, heterogeeneous, 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

μse (after MUltiparty SEssions)

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

Types for

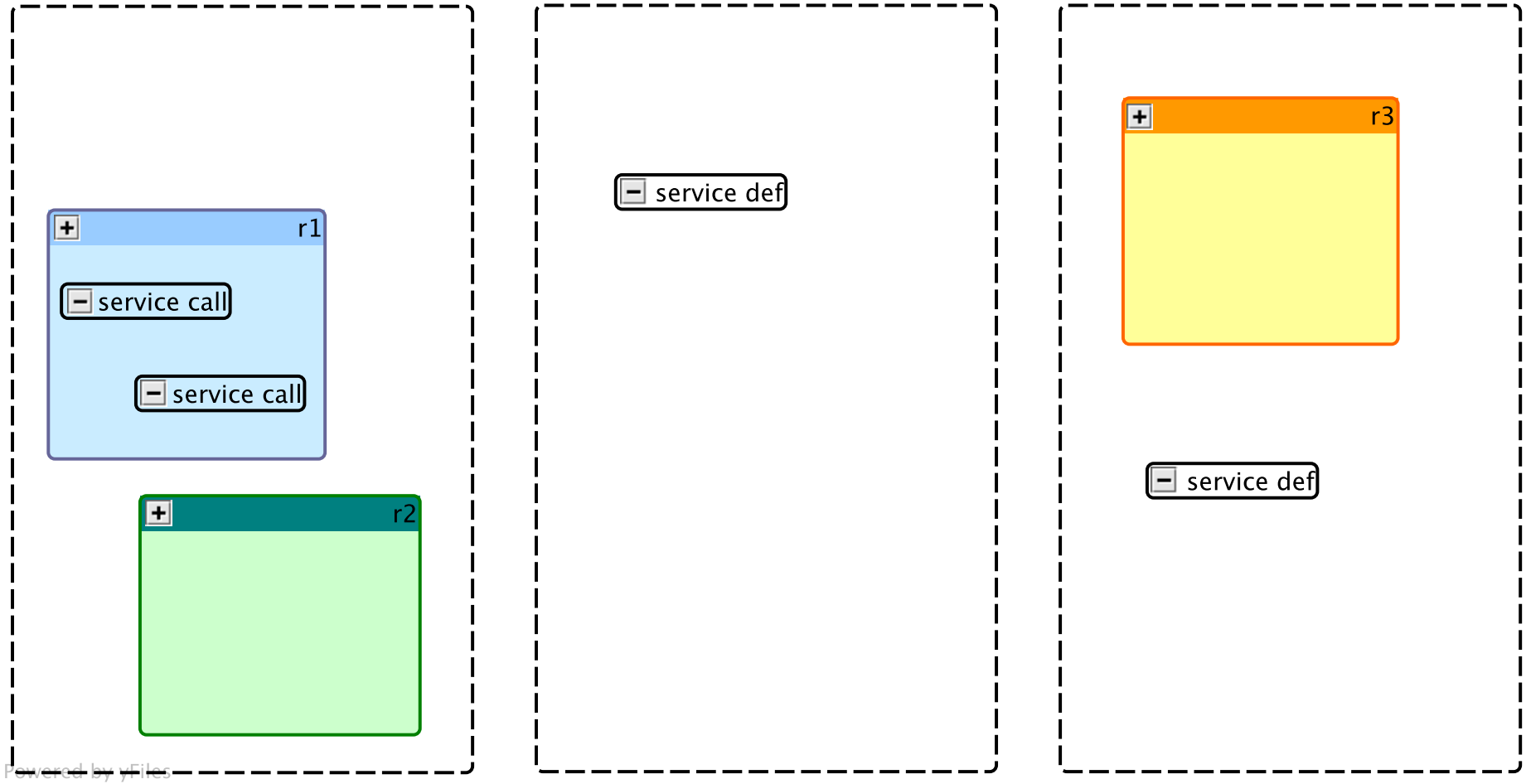
- Semantic description of services (for discovery)
- Compatibility check (for dynamic assembly)
- Early detection of possible sources of problems (trustworthiness)

Disclaim

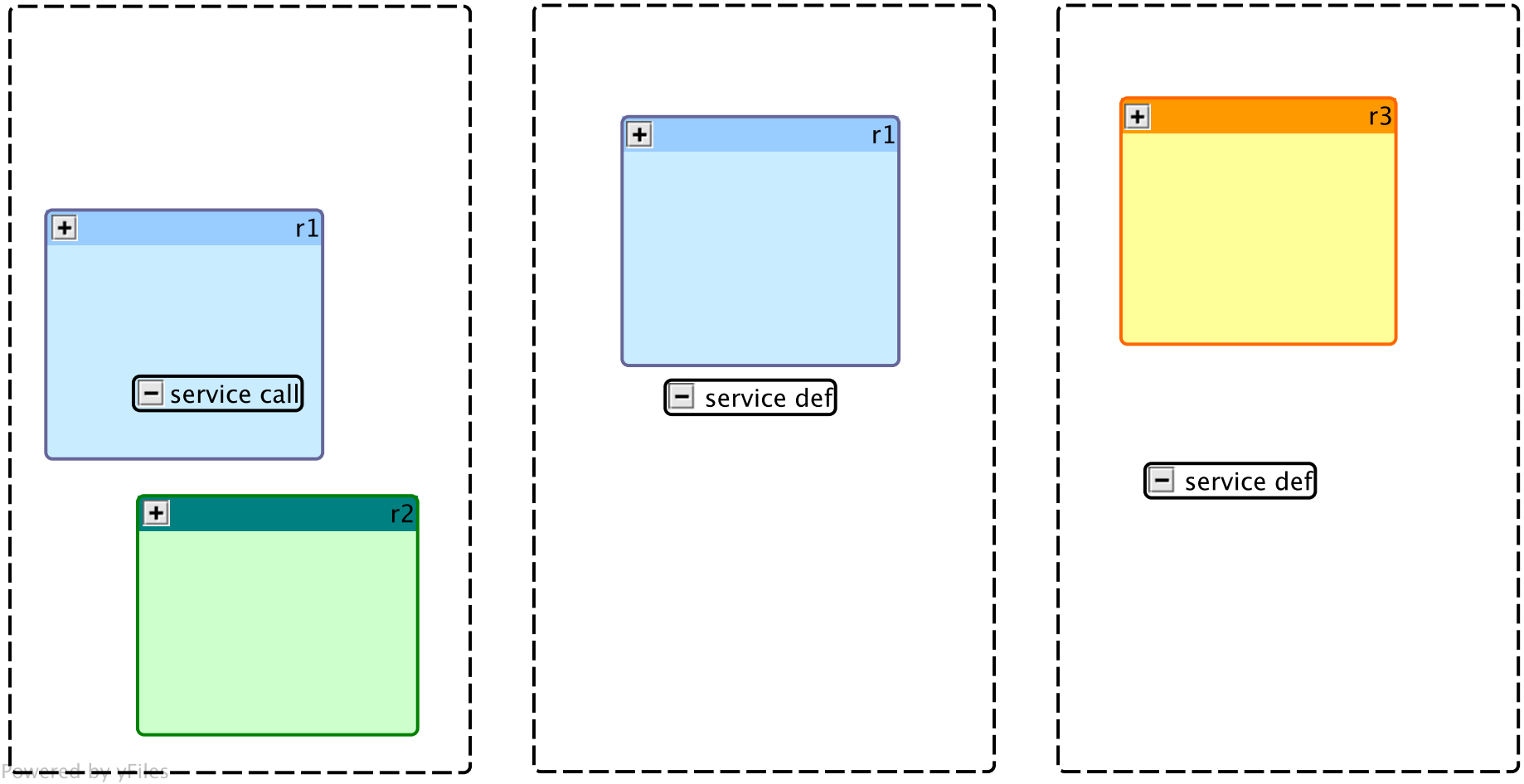
- We restrict to consider a "bare bones" fragment of μse
- We present a parametric type system w.r.t. 3 notions (task separation, dual type compatibility, session completion)
- We conjecture subject reduction + all non-typeable processes can deadlock
- We look for stronger guarantees

Outline

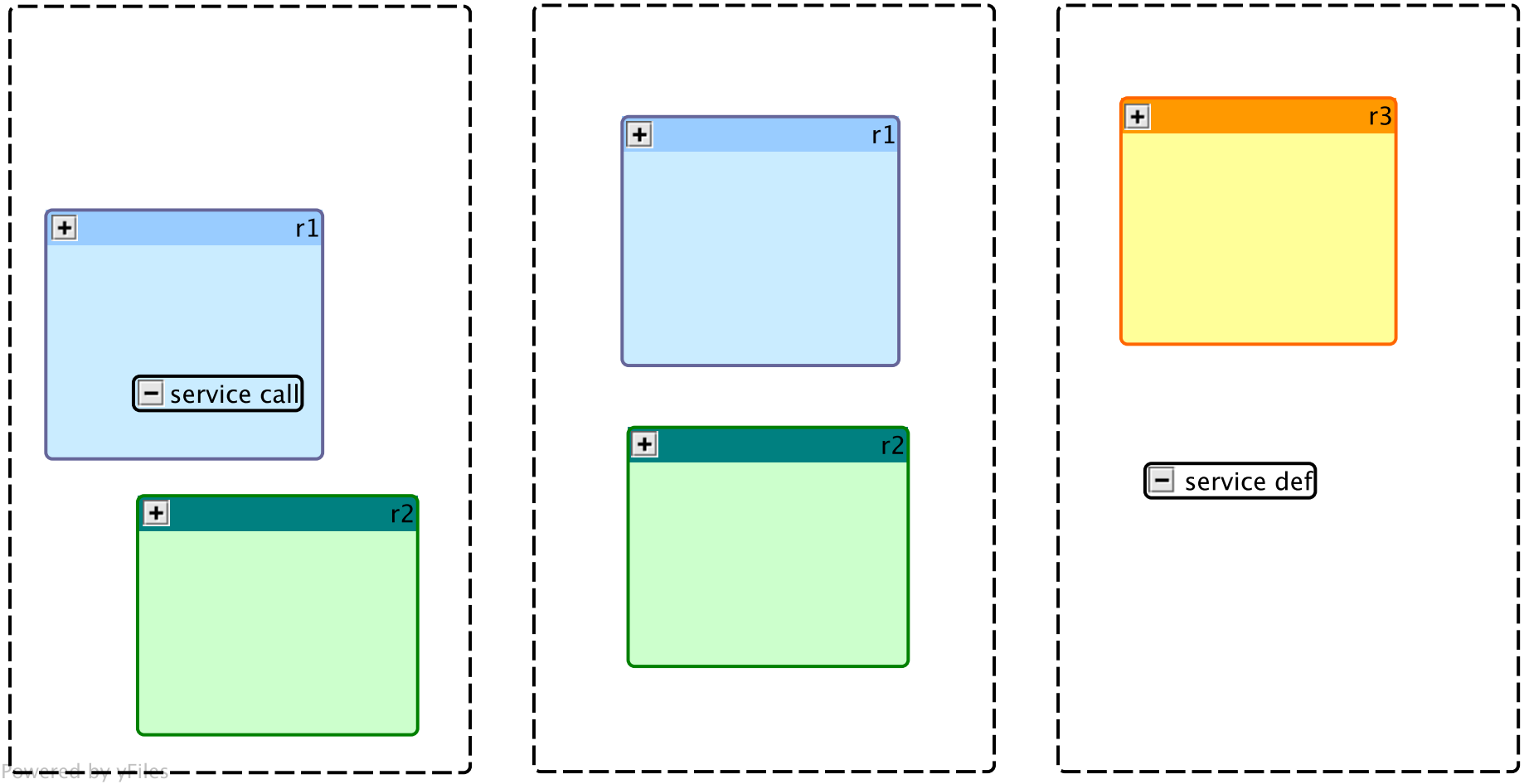
Sites, Service Invocation and Sessions



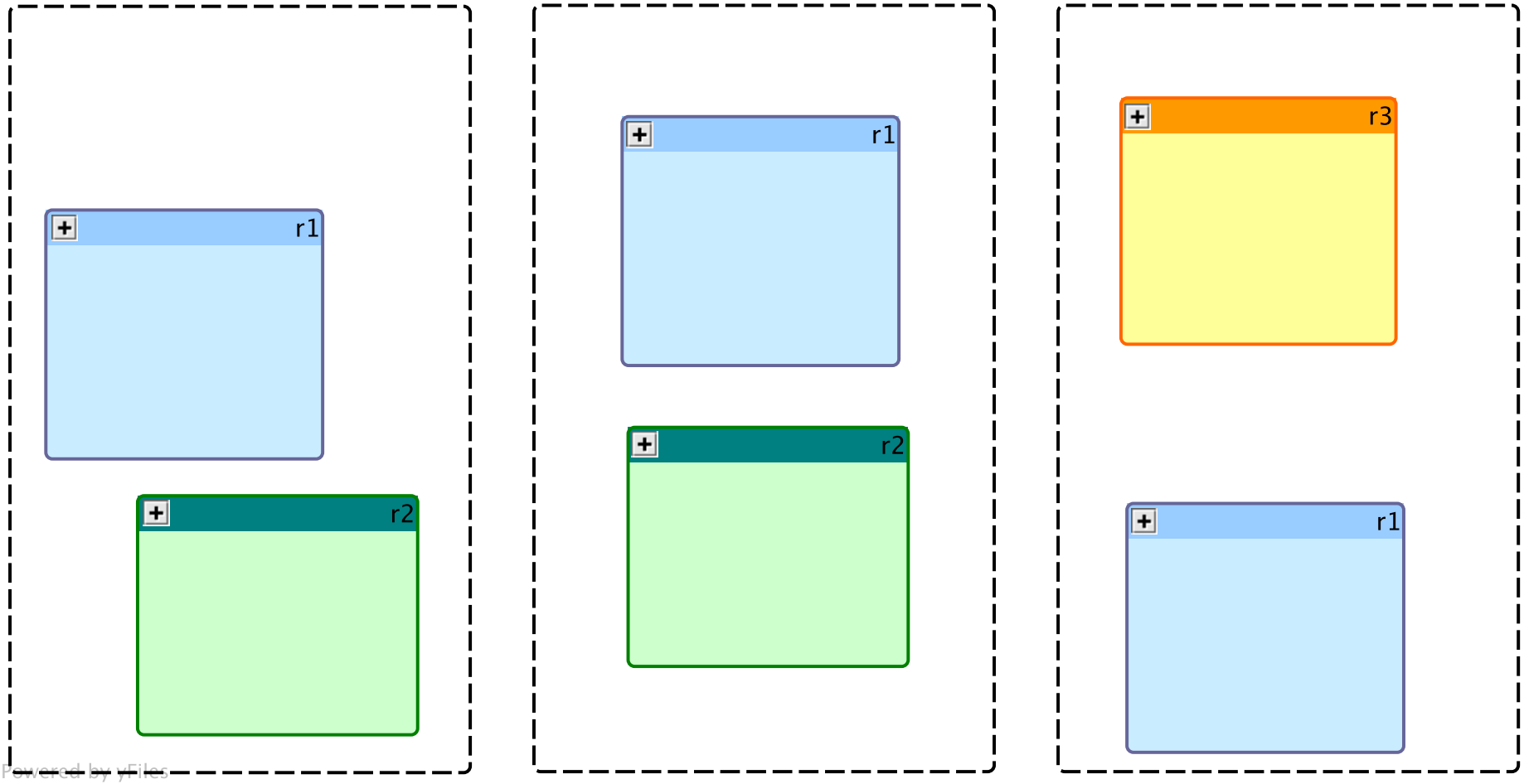
Sites, Service Invocation and Sessions



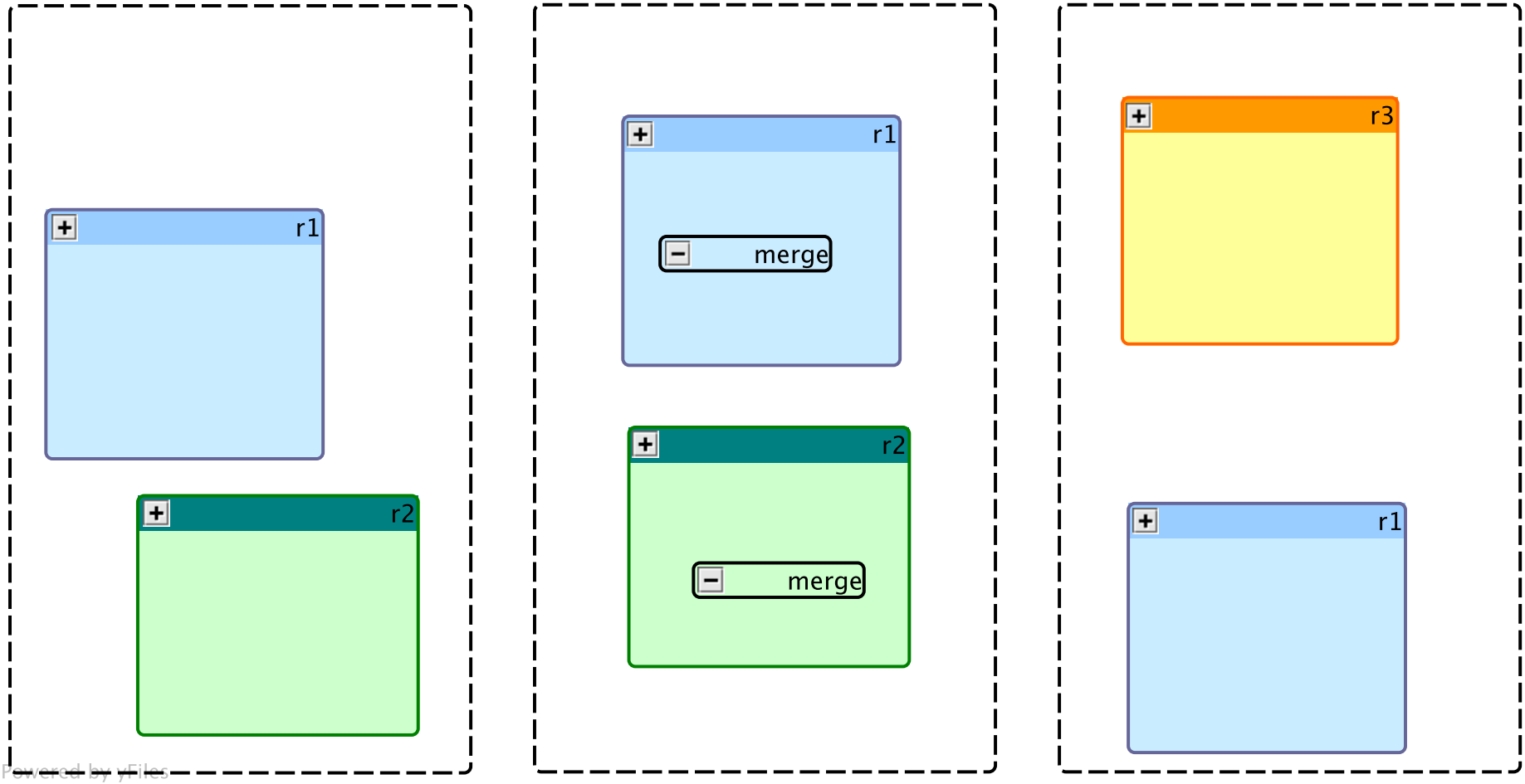
Sites, Service Invocation and Sessions



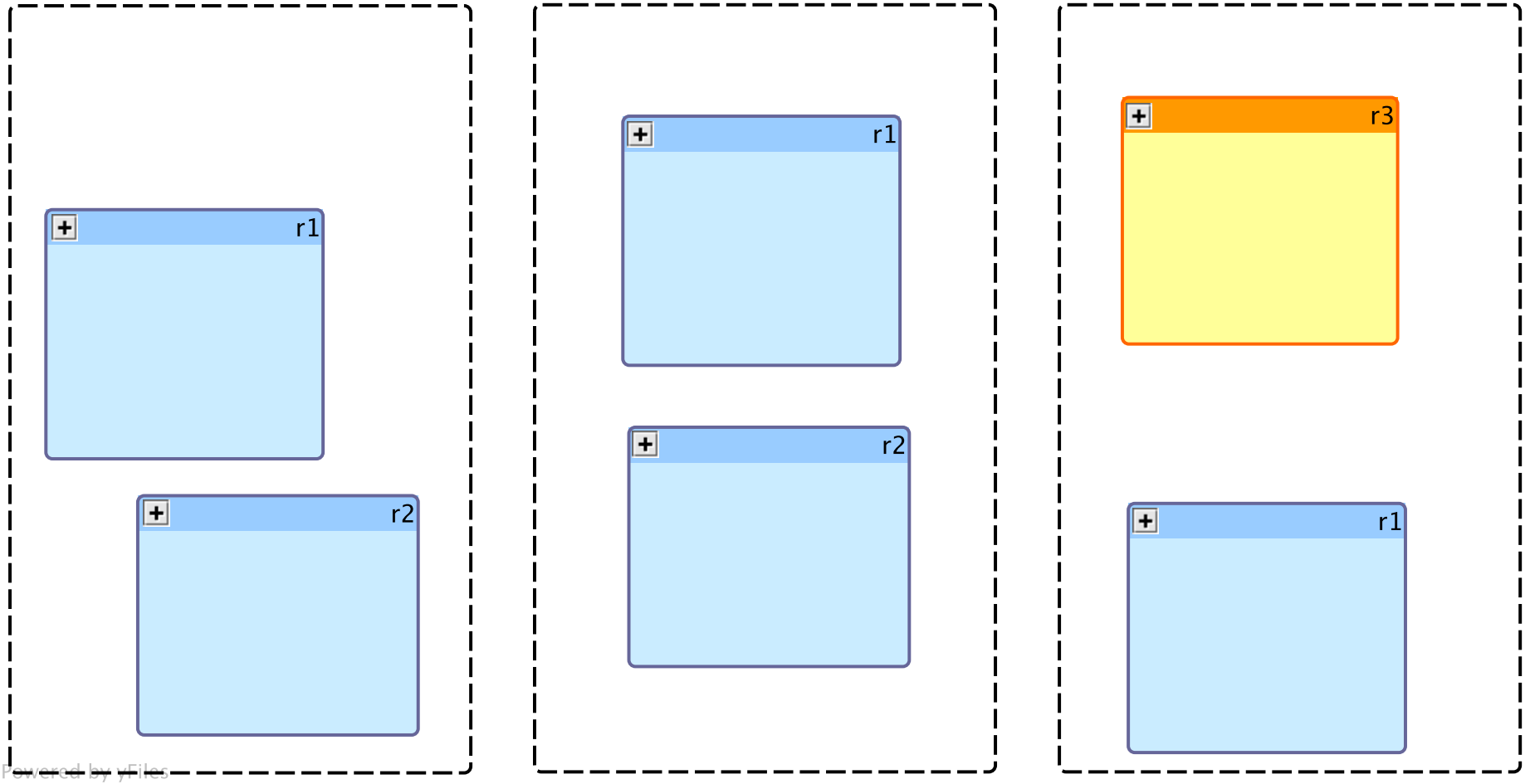
Sites, Service Invocation and Sessions



Merging Sessions



Merging Sessions



Outline

Types for Dynamic Multiparty Sessions

Type judgments

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

- we call σ the *current* type, and ρ the *delegated* type
- P provides communication activities in σ and ρ
- activities σ concern the current participants of its session,
- activities ρ concern other endpoints that P itself will allow to join its session (via service invocation or merge)
- Γ 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 ρ are delegated to $n^{\overline{p}}$.
- Δ is a finite partial mapping from session names r to types σ, 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

$$\frac{\Gamma; \Delta \vdash P : \{\sigma \nearrow \rho\}}{\Gamma; \Delta \vdash c.P : \{c * \sigma \nearrow \rho\}} \text{(TACTION)}$$

Parametric on Type Compatibility

Type compatibility

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

Our choice

Let $I(\sigma) = \{c \mid \exists \sigma' : \sigma \overset{c}{\mapsto} \sigma'\}$ be the set of initial actions of σ.
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 σ' and ρ' such that $\sigma \overset{x}{\mapsto} \sigma'$ and $\rho \overset{\overline{x}}{\mapsto} \rho'$, then $\sigma' \approx \rho'$.

Used in

Γ is *well-formed* if:

- whenever $\Gamma(n^p) = \sigma \nearrow \rho$, then $\Gamma(n^{\overline{p}}) = \sigma' \nearrow \rho'$ 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 σ that express admissible interactions of multiple endpoints.

Our choice

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

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

Used in

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

Example: Two Buyers

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

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

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*.

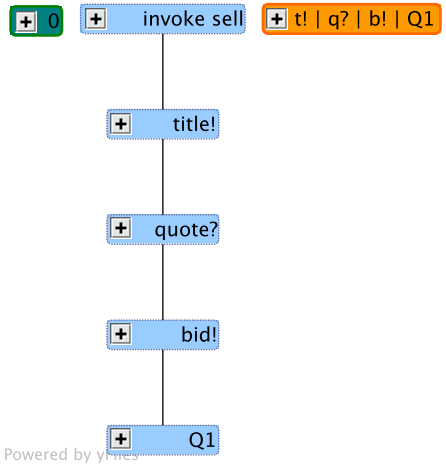
$$Offer = offer \Rightarrow \mathsf{merge}^{+}\ e.(\overline{title}.(\overline{quote}.Q_3 | \overline{quote}.Q_4))$$

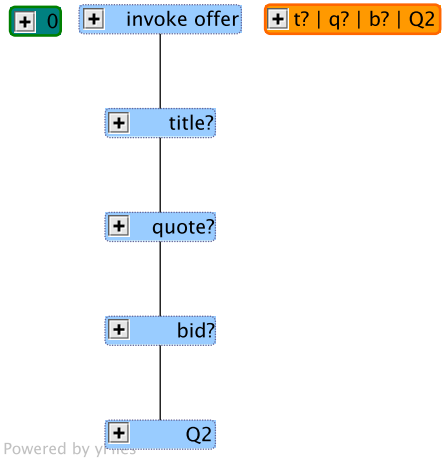
offer provides the book's *title* so that quotes are communicated to both buyers after the sessions are merged.

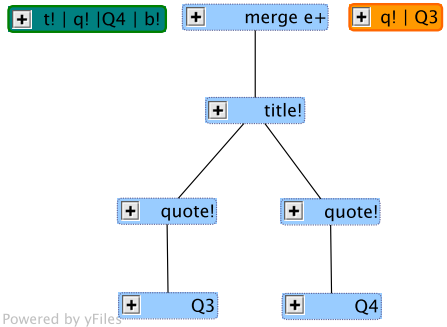
$$\begin{aligned} Buy_1 &= r_1 \triangleright \mathsf{invoke}\ sell.\overline{title}.quote.\overline{bid}.Q_1 \\ Buy_2 &= r_2 \triangleright \mathsf{invoke}\ offer.title.quote.bid.Q_2 \end{aligned}$$

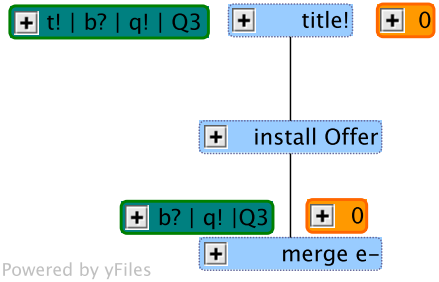
Buyers communicate over *bid* and the negotiation is concluded by the interactions among Q, Q_1 and Q_2 (not modeled here).

Example: Typing the Two Buyers









$$\Gamma = \{ \quad sell^+ : (0 \nearrow b|t|\overline{q}|Q_3), \quad sell^- : (0 \nearrow \overline{b}|\overline{t}.q|Q_1),$$
$$offer^+ : (0 \nearrow \overline{b}|\overline{t}|\overline{q}|Q_4), \quad offer^- : (0 \nearrow b|t|q|Q_2),$$
$$e^- : (\overline{b} \nearrow 0), \quad e^+ : (\overline{q}|b|Q_3 \nearrow 0) \ \}$$

(TINSTALL)

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

(TMERGE)

$$\frac{\Gamma;\Delta \vdash P : \{\sigma_1|\sigma_2|\sigma_3 \nearrow \rho\} \quad \Gamma(e^p) = \sigma|\sigma_2 \nearrow \sigma_3 \quad \Gamma(e^{\overline{p}}) = \sigma'|\sigma'' \nearrow \rho' \quad \sigma \approx \sigma''}{\Gamma;\Delta \vdash \mathsf{merge}^p\ e.P : \{\sigma'|\sigma_1|\sigma'' \nearrow \sigma_2|\sigma_3|\rho\}}$$

Outline

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!!

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!!