Provably correct implementations of services

Roberto Bruni[1] Rocco De Nicola[2] Michele Loreti[2]
Leonardo G. Mezzina[3]

[1]Dipartimento di Informatica, Università di Pisa, Italy
[2]Dipartimento di Sistemi e Informatica, Università di Firenze, Italy
[3]IMT Alti Studi Lucca, Italy

TGC 2008 — Barcelona, November, 3-4

Outline...

Outline...

Motivations...

The explosive growth of the Web has led to

- the widespread use of *communication centric* applications, often referred as *Web Services*;
- the growth of a new computational paradigm known as *Service Oriented Computing (SOC)*.

Motivations...

The explosive growth of the Web has led to

- the widespread use of *communication centric* applications, often referred as *Web Services*;
- the growth of a new computational paradigm known as *Service Oriented Computing (SOC)*.

- Service Oriented Computing (SOC) is calling for novel computational models and languages with primitives for client-server interaction, orchestration and unexpected events handling
- Important features of SOC are: compositionality, context-independence, encapsulation and re-usability.

Motivations...

The explosive growth of the Web has led to

- the widespread use of *communication centric* applications, often referred as *Web Services*;
- the growth of a new computational paradigm known as *Service Oriented Computing (SOC)*.

- Service Oriented Computing (SOC) is calling for novel computational models and languages with primitives for client-server interaction, orchestration and unexpected events handling
- Important features of SOC are: compositionality, context-independence, encapsulation and re-usability.

A number of formalisms have been defined to support the specification and analysis of service oriented applications at the right level of abstraction

Motivations...

These formalisms are based on process algebras enriched with primitives specific of service orientation:

- operators for manipulating semi-structured data
- mechanisms for describing safe client-service interactions
- constructors for composing possibly unreliable services
- techniques for query and discovery of services.

Motivations...

These formalisms are based on process algebras enriched with primitives specific of service orientation:

- operators for manipulating semi-structured data
- mechanisms for describing safe client-service interactions
- constructors for composing possibly unreliable services
- techniques for query and discovery of services.

A key point for the usefulness of process calculi is the availability of tools (types or logics) to specify, check and guarantee the correct behavior of the considered services.

Motivations...

We have defined a *Service Oriented Abstract Machine (SOAM)*...

- equipped with a formal semantics
- that can be used to implement the service specification formalisms.

Motivations...

We have defined a *Service Oriented Abstract Machine (SOAM)*...

- equipped with a formal semantics
- that can be used to implement the service specification formalisms.

The operational semantics of SOAM can be used as the basis for guaranteeing that the properties that have been proved by reasoning on the calculi-based specification are preserved by the actual implementations.

Motivations...

We have defined a *Service Oriented Abstract Machine (SOAM)*...

- equipped with a formal semantics
- that can be used to implement the service specification formalisms.

The operational semantics of SOAM can be used as the basis for guaranteeing that the properties that have been proved by reasoning on the calculi-based specification are preserved by the actual implementations.

Three representative service-oriented calculi will be considered.

Outline...

SOAM: Service Oriented Abstract Machine

SOAM is based on the notion of *queues*:

- model persistent, protected, communication lines;
- permit inter-task communication;
- are created on service invocation;
- messages are retrieved by means of *pattern matching*;
- can be either *synchronous* or *asynchronous*;
- naturally corresponds to the concept of session.

SOAM: Service Oriented Abstract Machine

SOAM network...

...can be:

- $\langle \sigma \vdash \mathcal{C} \rangle$, a program $\mathcal{C}$ running with local store σ
  - σ associates variable to values;
- $r : h$, a queue r with associated a sequence of values h;
- $\mathcal{N} | \mathcal{M}$, the parallel composition of two networks.

SOAM: Service Oriented Abstract Machine

SOAM network...

...can be:

- $\langle \sigma \vdash \mathcal{C} \rangle$, a program $\mathcal{C}$ running with local store σ
  - σ associates variable to values;
- $r : h$, a queue r with associated a sequence of values h;
- $\mathcal{N} | \mathcal{M}$, the parallel composition of two networks.

SOAM programs...

...are built from:

- standard imperative commands (iteration, selection,...);
- primitives for queues (creation, input and output);
- service definitions and invocations.

SOAM: Service Oriented Abstract Machine

Queue actions: `out`, `in`

(MOUT)

$$\frac{\sigma(w) = r}{\langle \sigma \vdash \mathtt{out}(w, \tilde{v}); \mathcal{C} \rangle | r : h \rightarrow \langle \sigma \vdash \mathcal{C} \rangle | r : \tilde{v} \cdot h}$$

(MIN)

$$\frac{\sigma(w) = r \quad \mathsf{match}(\sigma, \tilde{\mathcal{F}}_k, \tilde{v}) = \rho}{\langle \sigma \vdash \mathtt{in}(w, \Sigma_{j \in J}(\tilde{\mathcal{F}}_j.\mathcal{C}_j)); \mathcal{D} \rangle | r : h \cdot \tilde{v} \cdot h' \rightarrow \langle \sigma\rho \vdash \mathcal{C}_k; \mathcal{D} \rangle | r : h \cdot h'}$$

SOAM: Service Oriented Abstract Machine

Queue actions: `out`, `in`

$$\text{(MOUT)}\quad \frac{\sigma(w) = r}{\langle \sigma \vdash \mathtt{out}(w, \tilde{v}); \mathcal{C} \rangle | r : h \rightarrow \langle \sigma \vdash \mathcal{C} \rangle | r : \tilde{v} \cdot h}$$

$$\text{(MIN)}\quad \frac{\sigma(w) = r \quad \mathsf{match}(\sigma, \tilde{\mathcal{F}}_k, \tilde{v}) = \rho}{\langle \sigma \vdash \mathtt{in}(w, \Sigma_{j \in J}(\tilde{\mathcal{F}}_j.\mathcal{C}_j)); \mathcal{D} \rangle | r : h \cdot \tilde{v} \cdot h' \rightarrow \langle \sigma\rho \vdash \mathcal{C}_k; \mathcal{D} \rangle | r : h \cdot h'}$$

SOAM: Service Oriented Abstract Machine

Queue actions: `out`, `in`

(MOUT)

$$\frac{\sigma(w) = r}{\langle \sigma \vdash \mathtt{out}(w, \tilde{v}); \mathcal{C} \rangle | r : h \rightarrow \langle \sigma \vdash \mathcal{C} \rangle | r : \tilde{v} \cdot h}$$

(MIN)

$$\frac{\sigma(w) = r \quad \mathsf{match}(\sigma, \tilde{\mathcal{F}}_k, \tilde{v}) = \rho}{\langle \sigma \vdash \mathtt{in}(w, \Sigma_{j \in J}(\tilde{\mathcal{F}}_j.\mathcal{C}_j)); \mathcal{D} \rangle | r : h \cdot \tilde{v} \cdot h' \rightarrow \langle \sigma\rho \vdash \mathcal{C}_k; \mathcal{D} \rangle | r : h \cdot h'}$$

SOAM: Service Oriented Abstract Machine

Queue actions: `out`, `in`

$$(\textsc{Mout})\quad \frac{\sigma(w) = r}{\langle \sigma \vdash \mathtt{out}(w, \tilde{v}); \mathcal{C}\rangle | r : h \rightarrow \langle \sigma \vdash \mathcal{C}\rangle | r : \tilde{v} \cdot h}$$

$$(\textsc{Min})\quad \frac{\sigma(w) = r \quad \mathsf{match}(\sigma, \tilde{\mathcal{F}}_k, \tilde{v}) = \rho}{\langle \sigma \vdash \mathtt{in}(w, \Sigma_{j\in J}(\tilde{\mathcal{F}}_j.\mathcal{C}_j)); \mathcal{D}\rangle | r : h \cdot \tilde{v} \cdot h' \rightarrow \langle \sigma\rho \vdash \mathcal{C}_k; \mathcal{D}\rangle | r : h \cdot h'}$$

Queue creation: `new`

$$(\textsc{MnewR})\quad \frac{r \text{ is fresh}}{\langle \sigma \vdash \mathtt{new}\ x; \mathcal{C}\rangle \rightarrow (\nu r)(\langle \sigma[{}^{r}/_{x}] \vdash \mathcal{C}\rangle | r : \emptyset)}$$

SOAM: Service Oriented Abstract Machine

Queue actions: `out`, `in`

$$\text{(MOUT)}\quad \frac{\sigma(w) = r}{\langle \sigma \vdash \mathtt{out}(w, \tilde{v}); \mathcal{C}\rangle | r : h \rightarrow \langle \sigma \vdash \mathcal{C}\rangle | r : \tilde{v} \cdot h}$$

$$\text{(MIN)}\quad \frac{\sigma(w) = r \quad \mathsf{match}(\sigma, \tilde{\mathcal{F}}_k, \tilde{v}) = \rho}{\langle \sigma \vdash \mathtt{in}(w, \Sigma_{j\in J}(\tilde{\mathcal{F}}_j.\mathcal{C}_j)); \mathcal{D}\rangle | r : h \cdot \tilde{v} \cdot h' \rightarrow \langle \sigma\rho \vdash \mathcal{C}_k; \mathcal{D}\rangle | r : h \cdot h'}$$

Queue creation: `new`

$$\text{(MNEWR)}\quad \frac{r \text{ is fresh}}{\langle \sigma \vdash \mathtt{new}\ x; \mathcal{C}\rangle \rightarrow (\nu r)(\langle \sigma[^r/_x] \vdash \mathcal{C}\rangle | r : \emptyset)}$$

SOAM: Service Oriented Abstract Machine

Queue actions: out, in

$$\text{(MOUT)}\quad \frac{\sigma(w) = r}{\langle \sigma \vdash \mathtt{out}(w, \tilde{v}); \mathcal{C}\rangle | r : h \rightarrow \langle \sigma \vdash \mathcal{C}\rangle | r : \tilde{v} \cdot h}$$

$$\text{(MIN)}\quad \frac{\sigma(w) = r \quad \mathsf{match}(\sigma, \tilde{\mathcal{F}}_k, \tilde{v}) = \rho}{\langle \sigma \vdash \mathtt{in}(w, \Sigma_{j\in J}(\tilde{\mathcal{F}}_j.\mathcal{C}_j)); \mathcal{D}\rangle | r : h \cdot \tilde{v} \cdot h' \rightarrow \langle \sigma\rho \vdash \mathcal{C}_k; \mathcal{D}\rangle | r : h \cdot h'}$$

Queue creation: new

$$\text{(MNEWR)}\quad \frac{r \text{ is fresh}}{\langle \sigma \vdash \mathtt{new}\ x; \mathcal{C}\rangle \rightarrow (\nu r)(\langle \sigma[{}^{r}/_{x}] \vdash \mathcal{C}\rangle | r : \emptyset)}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]$$

$$\frac{}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]$$

$$\frac{}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1 \rangle, \mathcal{C}_1); \mathcal{D} \rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2 \rangle, \mathcal{C}_2); \mathcal{D}' \rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D} \rangle | \langle \sigma_2 \vdash \mathcal{D}' \rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1 \rho_1 \vdash \mathcal{C}_1 \rangle | \langle \sigma_2 \rho_2 \vdash \mathcal{C}_2 \rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]$$

$$\frac{}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]$$

$$\frac{}{\begin{array}{l}\langle \sigma_1 \vdash \mathtt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \mathtt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]$$

$$\frac{}{\begin{array}{l}\langle \sigma_1 \vdash \mathtt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \mathtt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \mathtt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \mathtt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

Task activation: `fork`

(MFORK)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\langle \sigma \vdash \mathtt{fork}(\langle x_1, y_1\rangle, \mathcal{C}_1, \langle x_2, y_2\rangle, \mathcal{C}_2)\rangle \rightarrow \begin{array}{l}(\nu r)(\nu r')(\langle \sigma\rho_1 \vdash \mathcal{C}_1\rangle | \\ \langle \sigma\rho_2 \vdash \mathcal{C}_2\rangle | r : \emptyset | r' : \emptyset)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

Task activation: `fork`

(MFORK)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\langle \sigma \vdash \texttt{fork}(\langle x_1, y_1\rangle, \mathcal{C}_1, \langle x_2, y_2\rangle, \mathcal{C}_2)\rangle \rightarrow \begin{array}{l}(\nu r)(\nu r')(\langle \sigma\rho_1 \vdash \mathcal{C}_1\rangle | \\ \langle \sigma\rho_2 \vdash \mathcal{C}_2\rangle | r : \emptyset | r' : \emptyset)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

Task activation: `fork`

(MFORK)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\langle \sigma \vdash \texttt{fork}(\langle x_1, y_1\rangle, \mathcal{C}_1, \langle x_2, y_2\rangle, \mathcal{C}_2)\rangle \rightarrow \begin{array}{l}(\nu r)(\nu r')(\langle \sigma\rho_1 \vdash \mathcal{C}_1\rangle | \\ \langle \sigma\rho_2 \vdash \mathcal{C}_2\rangle | r : \emptyset | r' : \emptyset)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \mathtt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \mathtt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

Task activation: `fork`

(MFORK)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\langle \sigma \vdash \mathtt{fork}(\langle x_1, y_1\rangle, \mathcal{C}_1, \langle x_2, y_2\rangle, \mathcal{C}_2)\rangle \rightarrow \begin{array}{l}(\nu r)(\nu r')(\langle \sigma\rho_1 \vdash \mathcal{C}_1\rangle | \\ \langle \sigma\rho_2 \vdash \mathcal{C}_2\rangle | r : \emptyset | r' : \emptyset)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\begin{array}{l}\langle \sigma_1 \vdash \texttt{offer}(a, \langle x_1, y_1\rangle, \mathcal{C}_1); \mathcal{D}\rangle | \\ \langle \sigma_2 \vdash \texttt{invoke}(a, \langle x_2, y_2\rangle, \mathcal{C}_2); \mathcal{D}'\rangle\end{array} \rightarrow \begin{array}{l}\langle \sigma_1 \vdash \mathcal{D}\rangle | \langle \sigma_2 \vdash \mathcal{D}'\rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1\rho_1 \vdash \mathcal{C}_1\rangle | \langle \sigma_2\rho_2 \vdash \mathcal{C}_2\rangle)\end{array}}$$

Task activation: `fork`

(MFORK)

$$\frac{r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]}{\langle \sigma \vdash \texttt{fork}(\langle x_1, y_1\rangle, \mathcal{C}_1, \langle x_2, y_2\rangle, \mathcal{C}_2)\rangle \rightarrow \begin{array}{l}(\nu r)(\nu r')(\langle \sigma\rho_1 \vdash \mathcal{C}_1\rangle | \\ \langle \sigma\rho_2 \vdash \mathcal{C}_2\rangle | r : \emptyset | r' : \emptyset)\end{array}}$$

SOAM: Service Oriented Abstract Machine

Service invocations and definitions: `invoke`, `offer`

(MSYNCH)

$$r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]$$

$$\begin{array}{l} \langle \sigma_1 \vdash \mathtt{offer}(a, \langle x_1, y_1 \rangle, \mathcal{C}_1); \mathcal{D} \rangle | \\ \langle \sigma_2 \vdash \mathtt{invoke}(a, \langle x_2, y_2 \rangle, \mathcal{C}_2); \mathcal{D}' \rangle \end{array} \rightarrow \begin{array}{l} \langle \sigma_1 \vdash \mathcal{D} \rangle | \langle \sigma_2 \vdash \mathcal{D}' \rangle | \\ (\nu r)(\nu r')(r : \emptyset | r' : \emptyset | \\ \quad \langle \sigma_1 \rho_1 \vdash \mathcal{C}_1 \rangle | \langle \sigma_2 \rho_2 \vdash \mathcal{C}_2 \rangle) \end{array}$$

Task activation: `fork`

(MFORK)

$$r, r' \text{ fresh} \quad \rho_i = [{}^{r}/_{x_i}][{}^{r'}/_{y_i}]$$

$$\langle \sigma \vdash \mathtt{fork}(\langle x_1, y_1 \rangle, \mathcal{C}_1, \langle x_2, y_2 \rangle, \mathcal{C}_2) \rangle \rightarrow \begin{array}{l} (\nu r)(\nu r')(\langle \sigma \rho_1 \vdash \mathcal{C}_1 \rangle | \\ \langle \sigma \rho_2 \vdash \mathcal{C}_2 \rangle | r : \emptyset | r' : \emptyset) \end{array}$$

SOAM: Service Oriented Abstract Machine

Syntax

$$
\begin{array}{lcll}
\mathcal{C}, \mathcal{D} & ::= & \texttt{skip} & \text{(skip)} \\
& | & \texttt{new } n & \text{(new)} \\
& | & \texttt{while } e \texttt{ do } \mathcal{C} & \text{(while)} \\
& | & \texttt{if } e \texttt{ then } \mathcal{C} \texttt{ else } \mathcal{D} & \text{(if-then-else)} \\
& | & \texttt{invoke}(v, \langle x, y \rangle, \mathcal{C}) & \text{(new session inv)} \\
& | & \texttt{offer}(v, \langle x, y \rangle, \mathcal{C}) & \text{(new session def)} \\
& | & \texttt{out}(w, \tilde{v}) & \text{(send)} \\
& | & \texttt{in}(w, \Sigma_{j \in J}(\tilde{\mathcal{F}}_j.\mathcal{C}_j)) & \text{(receive)} \\
& | & x := e & \text{(assignment)} \\
& | & \texttt{fork}(\langle x_1, y_1 \rangle, \mathcal{C}_1, \langle x_2, y_2 \rangle, \mathcal{C}_2) & \text{(fork and sync)} \\
& | & \mathcal{C}; \mathcal{D} & \text{(sequencing)} \\
\\
\mathcal{N} & ::= & \mathcal{O} & \text{(empty net)} \\
& | & \langle \sigma \vdash \mathcal{C} \rangle & \text{(running program)} \\
& | & \mathcal{N} | \mathcal{M} & \text{(network composition)} \\
& | & (\nu n)\mathcal{N} & \text{(name restriction)} \\
& | & r : h & \text{(session queue)}
\end{array}
$$

Outline...

Implementing Service Calculi with SOAM

Implementing Service Calculi with SOAM

Three service oriented calculi are considered:

- Session Language (SL);
- Calculus of Sessions and Pipelines (CaSPiS);
- and Orc.

Implementing Service Calculi with SOAM

Three service oriented calculi are considered:

- Session Language (SL);
- Calculus of Sessions and Pipelines (CaSPiS);
- and Orc.

These calculi provide specific primitives for modelling SOC: sessions (SL and CaSPiS), session delegation (SL), pipelining (Orc and CaSPiS), session nesting and pattern matching (CaSPiS), and cancelation of activities (Orc).

Implementing Service Calculi with SOAM

Three service oriented calculi are considered:

- Session Language (SL);
- Calculus of Sessions and Pipelines (CaSPiS);
- and Orc.

These calculi provide specific primitives for modelling SOC: sessions (SL and CaSPiS), session delegation (SL), pipelining (Orc and CaSPiS), session nesting and pattern matching (CaSPiS), and cancelation of activities (Orc).

For each of the above calculi we provide:

- a structural translation into the code of our abstract machine
- the operational correspondence between a process and its encoding.

Implementing Service Calculi with SOAM

Three service oriented calculi are considered:

- Session Language (SL);
- Calculus of Sessions and Pipelines (CaSPiS);
- and Orc.

These calculi provide specific primitives for modelling SOC: sessions (SL and CaSPiS), session delegation (SL), pipelining (Orc and CaSPiS), session nesting and pattern matching (CaSPiS), and cancelation of activities (Orc).

For each of the above calculi we provide:

- a structural translation into the code of our abstract machine
- the operational correspondence between a process and its encoding.

Calculus of Services with Pipelines and Sessions

Overview:

CaSPiS (*Calculus of Services with Pipelines and Sessions*) is a core calculus that relies on four three concepts:

Calculus of Services with Pipelines and Sessions

Overview:

CaSPiS (*Calculus of Services with Pipelines and Sessions*) is a core calculus that relies on four three concepts:

1. service definition/invocation

Calculus of Services with Pipelines and Sessions

Overview:

CaSPiS (*Calculus of Services with Pipelines and Sessions*) is a core calculus that relies on four three concepts:

1. service definition/invocation
2. bi-directional sessioning as a means for structuring client-service interaction

Calculus of Services with Pipelines and Sessions

Overview:

CaSPiS (*Calculus of Services with Pipelines and Sessions*) is a core calculus that relies on four three concepts:

1. service definition/invocation
2. bi-directional sessioning as a means for structuring client-service interaction
3. pipelining as a means of composing services.

CaSPiS in a nutshell...

CaSPiS in a nutshell...

Service definitions and invocations...

...are rendered respectively as $s.P$ and $\overline{s}.Q$:

- s is a service name
- P and Q implement the service and the client *protocols*

CaSPiS in a nutshell...

Service definitions and invocations...

...are rendered respectively as $s.P$ and $\overline{s}.Q$:

- s is a service name
- P and Q implement the service and the client *protocols*

$$\mathrm{news}.\langle \text{“}\textit{news item}\text{”} \rangle \mid \overline{\mathrm{news}}.(?x)\langle x \rangle^{\uparrow} Q$$

CaSPiS in a nutshell...

Service definitions and invocations...

...are rendered respectively as $s.P$ and $\overline{s}.Q$:

- s is a service name
- P and Q implement the service and the client *protocols*

$$\text{news}.\langle \text{“}\textit{news item}\text{”} \rangle \mid \overline{\text{news}}.(?x)\langle x \rangle^{\uparrow} Q$$

$$\downarrow$$

$$(\nu r)\left(r^{+} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle \mid r^{-} \triangleright (?x)\langle x \rangle^{\uparrow} Q\right)$$

CaSPiS in a nutshell...

Service definitions and invocations...

...are rendered respectively as $s.P$ and $\overline{s}.Q$:

- s is a service name
- P and Q implement the service and the client *protocols*

$$\mathsf{news}.\langle \text{“}\textit{news item}\text{”} \rangle \mid \overline{\mathsf{news}}.(?x)\langle x \rangle^{\uparrow} Q$$

$$\downarrow$$

$$(\nu r)\left(r^{+} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle \mid r^{-} \triangleright (?x)\langle x \rangle^{\uparrow} Q\right)$$

CaSPiS in a nutshell...

Service definitions and invocations...

...are rendered respectively as $s.P$ and $\overline{s}.Q$:

- s is a service name
- P and Q implement the service and the client *protocols*

$$\mathrm{news}.\langle \text{“}\textit{news item}\text{”} \rangle \mid \overline{\mathrm{news}}.(?x)\langle x \rangle^{\uparrow} Q$$

$$\downarrow$$

$$(\nu r)\left(r^{+} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle \mid r^{-} \triangleright (?x)\langle x \rangle^{\uparrow} Q\right)$$

We assume session be polarised to distinguish the two sides of a session (r^{+}, r^{-}):

- we let $\overline{r^{+}} = r^{-}$ and $\overline{r^{-}} = r^{+}$.

CaSPiS in a nutshell...

Abstractions and concretions...

Processes at the two sides of a session can interact with each other by means of:

- *concretions*: $\langle V \rangle$ sends value V over a session
- *abstractions*: $(\mathcal{F})P$ retrieves a value matching pattern $\mathcal{F}$.

CaSPiS in a nutshell...

Abstractions and concretions...

Processes at the two sides of a session can interact with each other by means of:

- *concretions*: $\langle V \rangle$ sends value V over a session
- *abstractions*: $(\mathcal{F})P$ retrieves a value matching pattern $\mathcal{F}$.

$$(\nu r)\left(r^{+} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle \mid r^{-} \triangleright (?x)\langle x \rangle^{\uparrow} Q\right)$$

CaSPiS in a nutshell...

Abstractions and concretions...

Processes at the two sides of a session can interact with each other by means of:

- *concretions*: $\langle V \rangle$ sends value V over a session
- *abstractions*: $(\mathcal{F})P$ retrieves a value matching pattern $\mathcal{F}$.

$$(\nu r)\left(r^{+} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle \mid r^{-} \triangleright (?x)\langle x \rangle^{\uparrow} Q\right)$$

CaSPiS in a nutshell...

Abstractions and concretions...

Processes at the two sides of a session can interact with each other by means of:

- *concretions*: $\langle V \rangle$ sends value V over a session
- *abstractions*: $(\mathcal{F})P$ retrieves a value matching pattern $\mathcal{F}$.

$$(\nu r)\left(r^{+} \triangleright \langle \text{“}news\ item\text{”} \rangle \mid r^{-} \triangleright (?x)\langle x \rangle^{\uparrow} Q\right)$$
$$\downarrow$$
$$(\nu r)\left(r^{+} \triangleright P \mid r^{-} \triangleright \langle \text{“}news\ item\text{”} \rangle^{\uparrow} Q\right)$$

CaSPiS in a nutshell...

Abstractions and concretions...

Processes at the two sides of a session can interact with each other by means of:

- *concretions*: $\langle V \rangle$ sends value V over a session
- *abstractions*: $(\mathcal{F})P$ retrieves a value matching pattern $\mathcal{F}$.

$$(\nu r)\left(r^{+} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle \mid r^{-} \triangleright (?x)\langle x \rangle^{\uparrow} Q\right)$$
$$\downarrow$$
$$(\nu r)\left(r^{+} \triangleright P \mid r^{-} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle^{\uparrow} Q\right)$$

Return...

Values can be returned outside a session to the enclosing environment using the return operator, $\langle \cdot \rangle^{\uparrow}$.

CaSPiS in a nutshell...

Abstractions and concretions...

Processes at the two sides of a session can interact with each other by means of:

- *concretions*: $\langle V \rangle$ sends value V over a session
- *abstractions*: $(\mathcal{F})P$ retrieves a value matching pattern $\mathcal{F}$.

$$(\nu r)\left(r^{+} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle \mid r^{-} \triangleright (?x)\langle x \rangle^{\uparrow} Q\right)$$
$$\downarrow$$
$$(\nu r)\left(r^{+} \triangleright P \mid r^{-} \triangleright \langle \text{“}\textit{news item}\text{”} \rangle^{\uparrow} Q\right)$$

Return...

Values can be returned outside a session to the enclosing environment using the return operator, $\langle \cdot \rangle^{\uparrow}$.

CaSPiS in a nutshell...

Pipeline...

Values returned by a session can be used to start new activities. This is achieved using the *pipeline* operator:

$$P > \tilde{x} > Q\,.$$

A *new* instance of process Q is activated each time P emits a value.

CaSPiS in a nutshell...

Pipeline...

Values returned by a session can be used to start new activities. This is achieved using the *pipeline* operator:

$$P > \tilde{x} > Q\,.$$

A *new* instance of process Q is activated each time P emits a value.

$$r \triangleright \langle \text{“}\mathit{news\ item}\text{”} \rangle^{\uparrow} Q > z > \overline{\mathsf{emailMe}}.\langle z \rangle$$

CaSPiS in a nutshell...

Pipeline...

Values returned by a session can be used to start new activities. This is achieved using the *pipeline* operator:

$$P > \tilde{x} > Q\,.$$

A *new* instance of process Q is activated each time P emits a value.

$$r \triangleright \langle \text{“}\textit{news item}\text{”} \rangle^{\uparrow} Q > z > \overline{\mathsf{emailMe}}.\langle z \rangle$$

CaSPiS in a nutshell...

Pipeline...

Values returned by a session can be used to start new activities. This is achieved using the *pipeline* operator:

$$P > \tilde{x} > Q\,.$$

A *new* instance of process Q is activated each time P emits a value.

$$r \triangleright \langle \text{“}news\ item\text{”} \rangle^{\uparrow} Q > z > \overline{\text{emailMe}}.\langle z \rangle$$

$$\downarrow$$

$$r \triangleright Q > z > \overline{\text{emailMe}}.\langle z \rangle \mid \overline{\text{emailMe}}.\langle \text{“}news\ item\text{”} \rangle$$

CaSPiS in a nutshell...

Pipeline...

Values returned by a session can be used to start new activities. This is achieved using the *pipeline* operator:

$$P > \tilde{x} > Q\,.$$

A *new* instance of process Q is activated each time P emits a value.

$$r \triangleright \langle \text{“}\textit{news item}\text{”} \rangle^{\uparrow} Q > z > \overline{\text{emailMe}}.\langle z \rangle$$

$$\downarrow$$

$$r \triangleright Q > z > \overline{\text{emailMe}}.\langle z \rangle \mid \overline{\text{emailMe}}.\langle \text{“}\textit{news item}\text{”} \rangle$$

Encoding CaSPiS in SOAM

Encoding CaSPiS in SOAM

The translation of CaSPiS relies on of two functions:

- net, that returns the SOAM network associated to a process P;
- prg, that returns the static program associated to a process.

Encoding CaSPiS in SOAM

The translation of CaSPiS relies on of two functions:

- net, that returns the SOAM network associated to a process P;
- prg, that returns the static program associated to a process.

Function net takes the references to the three queues used for identifying:

- the session used for retrieving *input* messages;
- the session used for delivering *output* messages;
- the session used for *returning* messages.

These queues are referenced in prg(P) by variables m_1^-, m_1^+ and m_2.

Encoding CaSPiS in SOAM

Sessions:

- A pair of queues (r^+, r^-) is associated to each session r:

Encoding CaSPiS in SOAM

Sessions:

- A pair of queues (r^+, r^-) is associated to each session r:

$$\mathsf{net}((\nu r)P, r_i, r_o, r_r) = (\nu r^+)(\nu r^-)(\mathsf{net}(P, r_i, r_o, r_r)|r^+ : \emptyset|r^- : \emptyset)$$

Encoding CaSPiS in SOAM

Sessions:

- A pair of queues (r^+, r^-) is associated to each session r:

$$\mathsf{net}((\nu r)P, r_i, r_o, r_r) = (\nu r^+)(\nu r^-)(\mathsf{net}(P, r_i, r_o, r_r)|r^+ : \emptyset|r^- : \emptyset)$$

Encoding CaSPiS in SOAM

Sessions:

- A pair of queues (r^+, r^-) is associated to each session r:

$$\mathsf{net}((\nu r)P, r_i, r_o, r_r) = (\nu r^+)(\nu r^-)(\mathsf{net}(P, r_i, r_o, r_r)|r^+ : \emptyset|r^- : \emptyset)$$

- Process executed within session r^p retrieves messages from $r^{\overline{p}}$, sends messages in r^p and returns values to r_o:

Encoding CaSPiS in SOAM

Sessions:

- A pair of queues (r^+, r^-) is associated to each session r:

$$\mathrm{net}((\nu r)P, r_i, r_o, r_r) = (\nu r^+)(\nu r^-)(\mathrm{net}(P, r_i, r_o, r_r)|r^+ : \emptyset|r^- : \emptyset)$$

- Process executed within session r^p retrieves messages from $r^{\overline{p}}$, sends messages in r^p and returns values to r_o:

$$\mathrm{net}(r^p \triangleright P, r_i, r_o, r_r) = \mathrm{net}(P, r_{\overline{p}}, r_p, r_o)$$

Encoding CaSPiS in SOAM

Sessions:

- A pair of queues (r^+, r^-) is associated to each session r:

$$\mathrm{net}((\nu r)P, r_i, r_o, r_r) = (\nu r^+)(\nu r^-)(\mathrm{net}(P, r_i, r_o, r_r)|r^+ : \emptyset|r^- : \emptyset)$$

- Process executed within session r^p retrieves messages from $r^{\overline{p}}$, sends messages in r^p and returns values to r_o:

$$\mathrm{net}(r^p \triangleright P, r_i, r_o, r_r) = \mathrm{net}(P, r_{\overline{p}}, r_p, r_o)$$

Encoding CaSPiS in SOAM

Sessions:

- A pair of queues (r^+, r^-) is associated to each session r:

$$\mathrm{net}((\nu r)P, r_i, r_o, r_r) = (\nu r^+)(\nu r^-)(\mathrm{net}(P, r_i, r_o, r_r)|r^+ : \emptyset|r^- : \emptyset)$$

- Process executed within session r^p retrieves messages from $r^{\overline{p}}$, sends messages in r^p and returns values to r_o:

$$\mathrm{net}(r^p \triangleright P, r_i, r_o, r_r) = \mathrm{net}(P, r_{\overline{p}}, r_p, r_o)$$

Encoding CaSPiS in SOAM

Service definition and invocation:

- These are directly mapped to SOAM service synchronization primitives where input and output queues are created after a synchronisation, while the return is performed in the current *out* queue:

$$\mathrm{net}(a.P, r_i, r_o, r_r) = \langle m_2 \mapsto r_o \vdash \texttt{offer}(a, \langle m_1^+, m_1^- \rangle, \mathrm{prg}(P)) \rangle$$

$$\mathrm{net}(\overline{a}.Q, r_i, r_o, r_r) = \langle m_2 \mapsto r_o \vdash \texttt{invoke}(a, \langle m_1^-, m_1^+ \rangle, \mathrm{prg}(Q)) \rangle$$

Encoding CaSPiS in SOAM

Service definition and invocation:

- These are directly mapped to SOAM service synchronization primitives where input and output queues are created after a synchronisation, while the return is performed in the current *out* queue:

$$\mathrm{net}(a.P, r_i, r_o, r_r) = \langle m_2 \mapsto r_o \vdash \texttt{offer}(a, \langle m_1^+, m_1^- \rangle, \mathrm{prg}(P)) \rangle$$

$$\mathrm{net}(\overline{a}.Q, r_i, r_o, r_r) = \langle m_2 \mapsto r_o \vdash \texttt{invoke}(a, \langle m_1^-, m_1^+ \rangle, \mathrm{prg}(Q)) \rangle$$

Encoding CaSPiS in SOAM

Service definition and invocation:

- These are directly mapped to SOAM service synchronization primitives where input and output queues are created after a synchronisation, while the return is performed in the current *out* queue:

$$\mathrm{net}(a.P, r_i, r_o, r_r) = \langle m_2 \mapsto r_o \vdash \texttt{offer}(a, \langle m_1^+, m_1^- \rangle, \mathrm{prg}(P)) \rangle$$

$$\mathrm{net}(\overline{a}.Q, r_i, r_o, r_r) = \langle m_2 \mapsto r_o \vdash \texttt{invoke}(a, \langle m_1^-, m_1^+ \rangle, \mathrm{prg}(Q)) \rangle$$

Encoding CaSPiS in SOAM

Pipeline:

- Values produced by P are stored in a fresh queue r_t. A process continuously retrieves values from r_t and executes a copy of Q:

$$\begin{array}{l}\mathsf{net}(P > \tilde{x} > Q, r_i, r_o, r_r) = \\ \quad (\nu r_t)(\mathsf{net}(P, r_i, r_t, r_r) | r_t : \emptyset \\ \quad \langle m_1^+ \mapsto r_o, m_1^- \mapsto r_i, m_2 \mapsto r_r \vdash \\ \quad\quad \texttt{while } \mathit{true} \texttt{ do in}(r_t, (\widetilde{?x}.\texttt{fork}(\mathsf{prg}(Q), \texttt{skip})))\rangle)\end{array}$$

Encoding CaSPiS in SOAM

Pipeline:

- Values produced by P are stored in a fresh queue r_t. A process continuously retrieves values from r_t and executes a copy of Q:

$$\begin{array}{l}\mathsf{net}(P > \tilde{x} > Q, r_i, r_o, r_r) = \\ \quad (\nu r_t)(\mathsf{net}(P, r_i, r_t, r_r) | r_t : \emptyset \\ \quad \langle m_1^+ \mapsto r_o, m_1^- \mapsto r_i, m_2 \mapsto r_r \vdash \\ \qquad \texttt{while } \mathit{true} \texttt{ do in}(r_t, (\widetilde{?x}.\texttt{fork}(\mathsf{prg}(Q), \texttt{skip})))\rangle)\end{array}$$

Encoding CaSPiS in SOAM

Pipeline:

- Values produced by P are stored in a fresh queue r_t. A process continuously retrieves values from r_t and executes a copy of Q:

$$\begin{array}{l}\mathsf{net}(P > \tilde{x} > Q, r_i, r_o, r_r) = \\ \quad (\nu r_t)(\mathsf{net}(P, r_i, r_t, r_r) | r_t : \emptyset \\ \quad \langle m_1^+ \mapsto r_o, m_1^- \mapsto r_i, m_2 \mapsto r_r \vdash \\ \qquad \texttt{while } \mathit{true} \texttt{ do } \texttt{in}(r_t, (\widetilde{?x}.\texttt{fork}(\texttt{prg}(Q), \texttt{skip})))\rangle)\end{array}$$

Encoding CaSPiS in SOAM

Pipeline:

- Values produced by P are stored in a fresh queue r_t. A process continuously retrieves values from r_t and executes a copy of Q:

$$\begin{array}{l}\mathsf{net}(P > \tilde{x} > Q, r_i, r_o, r_r) = \\ \quad (\nu r_t)(\mathsf{net}(P, r_i, r_t, r_r) | r_t : \emptyset \\ \quad \langle m_1^+ \mapsto r_o, m_1^- \mapsto r_i, m_2 \mapsto r_r \vdash \\ \qquad \texttt{while } \mathit{true} \texttt{ do in}(r_t, (\widetilde{?x}.\texttt{fork}(\mathsf{prg}(Q), \texttt{skip})))\rangle)\end{array}$$

Encoding CaSPiS in SOAM

Pipeline:

- Values produced by P are stored in a fresh queue r_t. A process continuously retrieves values from r_t and executes a copy of Q:

$$\begin{aligned}
&\mathsf{net}(P > \tilde{x} > Q, r_i, r_o, r_r) = \\
&\quad (\nu r_t)(\mathsf{net}(P, r_i, r_t, r_r) | r_t : \emptyset \\
&\quad \langle m_1^+ \mapsto r_o, m_1^- \mapsto r_i, m_2 \mapsto r_r \vdash \\
&\qquad \texttt{while } \mathit{true} \texttt{ do in}(r_t, \widetilde{(?x.\texttt{fork}(\mathsf{prg}(Q), \texttt{skip}))})\rangle)
\end{aligned}$$

Encoding CaSPiS in SOAM

Encoding CaSPiS in SOAM

Theorem (Completeness)

$$\textit{If } P \rightarrow Q \textit{ then } \mathsf{net}(P) \rightarrow^{*} \equiv \mathsf{net}(Q).$$

Encoding CaSPiS in SOAM

Theorem (Completeness)

If $P \rightarrow Q$ *then* $\mathsf{net}(P) \rightarrow^* \equiv \mathsf{net}(Q)$.

Theorem (Correctness)

If $\mathsf{net}(P) \rightarrow^* \mathcal{M}$ *then either* $\mathcal{M} \equiv \mathsf{net}(Q')$ *or there exists* $k > 0$ *s.t.* $\mathcal{M} \underbrace{\rightarrow \ldots \rightarrow}_{k} \equiv \mathsf{net}(Q')$ *and* $P \rightarrow^* Q$ *with* $Q \equiv Q'$.

Outline...

Conclusions...

- We have introduced SOAM, a service oriented abstract machine that can be used to implement service oriented calculi.
- SOAM provides *low-level* primitives for programming service oriented applications.
- Queues are used for modelling persistent and protected communication lines.

Conclusions...

- We have introduced SOAM, a service oriented abstract machine that can be used to implement service oriented calculi.
- SOAM provides *low-level* primitives for programming service oriented applications.
- Queues are used for modelling persistent and protected communication lines.

- We have used the proposed machine to implement three very different formalisms for service specification: the Session Language (SL), CaSPiS, and Orc.
- For all of them we have proved that the proposed implementation is operationally correct (sound and complete).

Future work...

- We plan to investigate the extensions that are needed to deal with more advanced features of service oriented computing such as
  - controlled service closures,
  - compensations,
  - multiparty synchronization.
- We plan to provide a complete prototype implementation of our machine.

Thank You