A Semantics

A.1 Behavioural Layer

$$\text{(P-Choice)}\ \frac{j\in J \quad \eta_j \xrightarrow{\mu} Q_j}{\sum_{i\in J}[\eta_i]\{P_i\} \xrightarrow{\mu} Q_j;P_j}$$

$$\text{(P-Solicit)}\ \texttt{o@p(p')(p'')} \xrightarrow{\boldsymbol{\nu} r\ \texttt{o@p(p')}} \mathsf{Wait}(r,\texttt{p''}) \qquad \text{(P-Notify)}\ \texttt{o@p(p')} \xrightarrow{\boldsymbol{\nu} r\ \texttt{o@p(p')}} \mathbf{0}$$

$$\text{(P-Req)}\ \texttt{o(p)(p')}\{P\} \xrightarrow{r:\texttt{o(p)}} \mathsf{Exec}(r,\texttt{p'},P) \qquad \text{(P-OneWay)}\ \texttt{o(p)} \xrightarrow{r:\texttt{o(p)}} \mathbf{0}$$

$$\text{(P-End-Exec)}\ \mathsf{Exec}(r,\texttt{p},\mathbf{0}) \xrightarrow{\overline{r}\,\texttt{p}} \mathbf{0} \qquad \text{(P-Wait)}\ \mathsf{Wait}(r,\texttt{p}) \xrightarrow{r\,\texttt{p}} \mathbf{0}$$

$$\text{(P-Exec)}\ \frac{P \xrightarrow{\mu} P'}{\mathsf{Exec}(r,\texttt{p},P) \xrightarrow{\mu} \mathsf{Exec}(r,\texttt{p},P')} \qquad \text{(P-Asgn)}\ \texttt{p}=e \xrightarrow{\texttt{p}=e} \mathbf{0}$$

$$\text{(P-Sequence)}\ \frac{P \xrightarrow{\mu} P'}{P;Q \xrightarrow{\mu} P';Q} \qquad \text{(P-Parallel)}\ \frac{P \xrightarrow{\mu} P'}{P \mid Q \xrightarrow{\mu} P' \mid Q}$$

$$\text{(P-If-Then)}\ \frac{e(t)=true}{\texttt{if}(e)\{P\}\,\texttt{else}\{Q\} \xrightarrow{\texttt{read}\ v} P} \qquad \text{(P-If-Else)}\ \frac{e(t)=false}{\texttt{if}(e)\{P\}\,\texttt{else}\{Q\} \xrightarrow{\texttt{read}\ v} Q}$$

$$\text{(P-Iteration)}\ \frac{e(t)=true}{\texttt{while}(e)\{P\} \xrightarrow{\texttt{read}\ v} P;\texttt{while}(e)\{P\}} \qquad \text{(P-No-Iteration)}\ \frac{e(t)=false}{\texttt{while}(e)\{P\} \xrightarrow{\texttt{read}\ v} \mathbf{0}}$$

A.2 Service Layer

The structure of the whole service is omitted where it is irrelevant.

$$\text{(S-Get)}\ \frac{P \xrightarrow{r:\texttt{o(p)}} P'}{P\cdot t\cdot(r,\texttt{o},t'):\tilde{m} \xrightarrow{\tau} P'\cdot t\leftarrow_{\texttt{p}} t'\cdot\tilde{m}} \qquad \text{(S-Send)}\ \frac{P \xrightarrow{\boldsymbol{\nu} r\ \texttt{o@p(p')}} P'}{P\cdot t\cdot\tilde{m} \xrightarrow{\boldsymbol{\nu} r\ \texttt{o@p}(t)^{\dagger}(\texttt{p'}(t))} P'\cdot t\cdot\tilde{m}}$$

$$\text{(S-SR)}\ \frac{P \xrightarrow{\overline{r}\,\texttt{p}} P'}{P\cdot t\cdot\tilde{m} \xrightarrow{\overline{r}\,\texttt{p}(t)} P'\cdot t\cdot\tilde{m}} \qquad \text{(S-RR)}\ \frac{P \xrightarrow{r\,\texttt{p}} P'}{P\cdot t\cdot\tilde{m} \xrightarrow{r\,t'} P'\cdot t\leftarrow_{\texttt{p}} t'\cdot\tilde{m}}$$

$$\text{(S-Asgn)}\ \frac{P \xrightarrow{\texttt{p}=e} P'}{P\cdot t\cdot\tilde{m} \xrightarrow{\tau} P'\cdot t\twoheadleftarrow_{\texttt{p}} e(t)\cdot\tilde{m}} \qquad \text{(S-Read)}\ \frac{P \xrightarrow{\texttt{read}\ v} P'}{P\cdot t\cdot\tilde{m} \xrightarrow{\tau} P'\cdot t\cdot\tilde{m}}$$

$$\text{(S-Corr)}\ \frac{t',\texttt{o} \vdash_{\alpha_C} t}{P\triangleright_{\alpha_C} I \mid P'\cdot t\cdot\tilde{m} \xrightarrow{\boldsymbol{\nu} r\ \texttt{o}(t')} P\triangleright_{\alpha_C} I \mid P'\cdot t\cdot\tilde{m}:(r,\texttt{o},t')}$$

$$\text{(S-Start)}\ \frac{t,\texttt{o} \nvdash_{\alpha_C} I \qquad P \xrightarrow{r:\texttt{o(p)}} P' \qquad t'=\mathsf{init}(t,\texttt{o},\alpha_C)}{P\triangleright_{\alpha_C} I \xrightarrow{\boldsymbol{\nu} r\ \texttt{o}(t)} P\triangleright_{\alpha_C} I \mid P'\cdot t_{\perp}\leftarrow_{\texttt{p}} t\leftarrow_{\texttt{csets}} t'\cdot\epsilon}$$

$$\mathsf{init}(t,\texttt{o},\alpha_C) = \begin{cases} t_{\perp} \twoheadleftarrow_{\texttt{p}_1} f(\texttt{p}_1)(t) \ldots \twoheadleftarrow_{\texttt{p}_n} f(\texttt{p}_n)(t) & \text{if } \alpha_C(\texttt{o}) = (\{\texttt{p}_1,\ldots,\texttt{p}_n\},f) \\ t_{\perp} & \text{if } \texttt{o} \notin Dom(\alpha_C) \\ \text{undefined} & \text{otherwise} \end{cases}$$

A.3 Network Layer

$$\text{(N-Comm)}\quad \frac{S_1 \xrightarrow{\nu r\, o@l_2(t_M)} S_1' \qquad S_2 \xrightarrow{\nu r\, o(t_M)} S_2' \qquad r \notin \mathtt{cn}(S_1) \cup \mathtt{cn}(S_2)}{[S_1]_{l_1} \mid [S_2]_{l_2} \xrightarrow{\tau} \nu r\,(\ [S_1']_{l_1} \mid [S_2']_{l_2}\ )}$$

$$\text{(N-Response)}\quad \frac{S_1 \xrightarrow{r\, t_M} S_1' \qquad S_2 \xrightarrow{\overline{r}\, t_M} S_2'}{\nu r\,(\ [S_1]_{l_1} \mid [S_2]_{l_2}\ ) \xrightarrow{\tau} [S_1']_{l_1} \mid [S_2']_{l_2}} \qquad \text{(N-Parallel)}\quad \frac{N_1 \xrightarrow{\mu} N_1'}{N_1 \mid N_2 \xrightarrow{\mu} N_1' \mid N_2}$$

$$\text{(N-Tau)}\quad \frac{S \xrightarrow{\tau} S'}{[S]_l \mid N \xrightarrow{\tau} [S']_l \mid N} \qquad \text{(N-Congruence)}\quad \frac{N_1 \equiv N_1' \qquad N_1' \xrightarrow{\mu} N_2' \qquad N_2' \equiv N_2}{N_1 \xrightarrow{\mu} N_2}$$

B Type system

(T-Nil) $\dfrac{}{\Gamma \vdash \mathbf{0} : \emptyset \mid \emptyset}$

(T-CSets-New) $\dfrac{}{\Gamma \vdash \texttt{csets.p=new} : \emptyset \mid \{\texttt{p}^{\circ}\}}$

(T-CSets-Expr) $\dfrac{e \text{ not undefined}}{\Gamma \vdash \texttt{csets.p=}e : \emptyset \mid \{\texttt{p}^{\bullet}\}}$

(T-Assign) $\dfrac{\texttt{p} \neq \texttt{csets.p}'}{\Gamma \vdash \texttt{p = } e : \emptyset \mid \emptyset}$

(T-If-Then-Else) $\dfrac{\Gamma \vdash P : \Delta_{N_1} \mid \Delta_P \qquad \Gamma \vdash Q : \Delta_{N_2} \mid \Delta_P}{\Gamma \vdash \texttt{if}(e)\,\{P\}\,\texttt{else}\,\{Q\} : \Delta_{N_1} \cup \Delta_{N_2} \mid \Delta_P}$

(T-Choice) $\dfrac{\forall j \in J \;.\; \Gamma \vdash \eta_j ; P_j : \Delta_{N_j} \mid \Delta_P}{\Gamma \vdash \sum_{i \in J} [\eta_i]\{P_i\} : \bigcup_{j \in J} \Delta_{N_j} \mid \Delta_P}$

(T-Par) $\dfrac{\Gamma \vdash P : \Delta_{N_1} \mid \Delta_{P_1} \qquad \Gamma \vdash Q : \Delta_{N_2} \mid \Delta_{P_2}}{\Gamma \vdash P \mid Q : \Delta_{N_1} \cup \Delta_{N_2} \mid \Delta_{P_1} \uplus \Delta_{P_2}}$

(T-Seq) $\dfrac{\Gamma \vdash P : \Delta_{N_1} \mid \Delta_{P_1} \qquad \Gamma \vdash Q : \Delta_{N_2} \mid \Delta_{P_2} \qquad \Delta' = (\Delta_{N_2} \setminus \Delta_{P_1}) \cup \Delta_{N_1}}{\Gamma \vdash P ; Q : \Delta' \mid \Delta_{P_1} \uplus \Delta_{P_2}}$

(T-While) $\dfrac{\Gamma \vdash P : \Delta_N \mid \emptyset}{\Gamma \vdash \texttt{while}(e)\,\{P\} : \Delta_N \mid \emptyset}$

(T-Notification) $\dfrac{}{\Gamma \vdash \texttt{o@p(p')} : \emptyset \mid \emptyset}$

(T-SolicitResponse) $\dfrac{\texttt{p''} \neq \texttt{csets.p'''}}{\Gamma \vdash \texttt{o@p(p')(p'')} : \emptyset \mid \emptyset}$

(T-OneWay) $\dfrac{\Gamma(\texttt{o}) = c \neq \emptyset \qquad \texttt{p} \neq \texttt{csets.p}'}{\Gamma \vdash \texttt{o(p)} : c \mid \emptyset}$

(T-OneWay-Start) $\dfrac{(\Gamma \vdash \texttt{o(p)} : c \mid \emptyset \;\vee\; c = \emptyset) \;\wedge\; \texttt{p} \neq \texttt{csets.p}'}{\Gamma \vdash_s \texttt{o(p)} : c \mid \emptyset}$

(T-RequestResponse) $\dfrac{\Gamma(o) = c \neq \emptyset \qquad \texttt{p} \neq \texttt{csets.p}' \qquad \Gamma \vdash P : \Delta_N \mid \Delta_P}{\Gamma \vdash \texttt{o(p)(p')}\,\{P\} : c \cup \Delta_N \mid \Delta_P}$

(T-RequestResponse-Start) $\dfrac{(\Gamma \vdash \texttt{o(p)(p')}\,\{P\} : c \mid \emptyset \;\vee\; c = \emptyset) \qquad \texttt{p} \neq \texttt{csets.p}' \qquad \Gamma \vdash P : \Delta_N \mid \Delta_P}{\Gamma \vdash_s \texttt{o(p)(p')}\,\{P\} : c \cup \Delta_N \mid \Delta_P}$

(T-Starter) $\dfrac{}{\Gamma \vdash \mathbf{0} \triangleright_{\alpha_C} P \cdot t_{\perp} \cdot \epsilon : \emptyset \mid \emptyset}$

(T-Service) $\dfrac{\forall j \in J. \begin{cases} \Gamma_j \vdash_s \eta_j : c_j \mid \Delta_{P_j} \;\wedge\; \Gamma_j \vdash P_j : \Delta'_{N_j} \mid \Delta'_{P_j} \;\wedge\; \Delta'_{N_j} \subseteq c_j \uplus \Delta_{P_j} \;\wedge \\ \forall \texttt{o} \in Dom(\Gamma_j).\, \alpha_C(\texttt{o}) = (\Gamma_j(\texttt{o}), f) \;\wedge\; Dom(f) = \Gamma_j(\texttt{o}) \;\wedge\; \Delta_{P_j} \uplus \Delta'_{P_j} \ltimes C \;\wedge \\ \forall i \in J.\; i \neq j \Rightarrow \Gamma_i \vdash_s \eta_i : c_i \mid \Delta_{P_i} \;\wedge\; \Gamma_i \vdash P_i : \Delta'_{N_i} \mid \Delta'_{P_i} \;\wedge\; (\Delta_{P_i} \cup \Delta_{P_j}) \cap c_j = \emptyset \end{cases}}{\emptyset \vdash \sum_{i \in J} [\eta_i]\{P_i\} \triangleright_{\alpha_C} I : \emptyset \mid \emptyset}$

B.1 Run-time types

$$\text{(T-Wait)}\quad \frac{}{\Gamma \vdash \mathsf{Wait}(r,\mathtt{p}) : \emptyset \mid \emptyset}$$

$$\text{(T-Exec)}\quad \frac{\Gamma \vdash P : \Delta_N \mid \Delta_P}{\Gamma \vdash \mathsf{Exec}(r,\mathtt{p},P) : \Delta_N \mid \Delta_P}$$

$$\text{(T-Session)}\quad \frac{\Gamma \vdash P : \Delta_N \mid \Delta_P \quad \forall \mathtt{p} \in \Delta_N.\mathtt{csets.p}(t) \neq v_\perp \quad \forall \mathtt{p} \in \Delta_P.\mathtt{csets.p}(t) = v_\perp}{\Gamma \vdash P \cdot t \cdot \tilde{m} : \emptyset \mid \emptyset}$$

$$\text{(T-Runtime-Service)}\quad \frac{\Gamma \vdash P : \Delta_N \mid \Delta_P \quad \forall j \in J.\Gamma \vdash P_j \cdot t_j \cdot \tilde{m}_j : \emptyset \mid \emptyset \quad \nexists j_1, j_2 \in J, c \in C.\forall \mathtt{p} \in c.\mathtt{csets.p}(t_{j_1}) = \mathtt{csets.p}(t_{j_2}) \neq v_\perp}{\Gamma \vdash P \triangleright_{\alpha_C} \prod_{j \in J} P_j \cdot t_j \cdot \tilde{m}_j : \emptyset \mid \emptyset}$$

B.2 Discussion

We report here an extended discussion on the type system.

The rules for typing One-Way and Request-Response, `T-OneWay` and `T-RequestResponse` input statements are used by rule `T-Service` in alternative forms, `T-OneWay-Start` and `T-RequestResponse-Start`, for checking the special case of an input statement used as a session guard. By definition of $\vdash$, if the correlation set for an operation is empty then no message can correlate for that operation. Therefore using an operation with empty associated correlation set inside a session is undesirable, because no message for that operation will be ever put in the session message queue. Rule `S-Start` does not need any correlation to hold for starting a session with a received message, so in that case we do not apply the same restriction.

Rule `T-CSets-Expr` requires expression e to yield a value that is not undefined. In other words, if t is the state of the session in which P is running, then $e(t) \neq v_\perp$. In the JOLIE language, the evaluation of an expression e is not $v_\perp$ if one of the following conditions applies:

- there is a constant value in e;
- there is a path `p` in e that points to a node in the session state that is not $v_\perp$.

Checking the first condition is trivially done by inspecting the structure of e. The second condition, instead, requires a definite assignments analysis similar to that performed for the C# and the Java languages. Our implementation follows the same principles and does not introduce any new aspect.

The type system guarantees that assignments to correlation variables are affine, i.e. once a correlation variable is instantiated its value can not be redefined. This ensures that once an external party gets to know a correlation value for communicating with a session, said value is valid until the session terminates. The BPEL language uses the same idea, treating correlation variables as late-bound constants.

Rules `T-OneWay` and `T-RequestResponse` require the correlation set used by the operation to be fully instantiated before executing the input statement. This check could be relaxed by allowing to wait for a parallel process to initialise the needed correlation variables.

The type system does not consider propagation of correlation value freshness. In assignments of the form `csets.p = csets.p'` if `p'` has a fresh value then also `p` will have a fresh value wrt other sessions. We did not find any convincing example for motivating this feature, because a programmer that wants to reuse a fresh value may already use a same correlation variable in more than one correlation set. The rules could be nonetheless extended to support this relaxed condition.

Another way for guaranteeing a correct initialization of correlation values would be to support the instantiation of an additional correlation set by means of inputs, similarly to what is done in rule `S-Start`, performed by running sessions. This approach would be similar to the one offered by the BPEL language through the `initiate` attribute for correlation sets in message receives. It is possible to extend our model with such a feature by adding auxiliary aliasing functions to the service definition, that will tell which correlation sets would be initialised by each different input. This would, however, put more burden on the semantics: the check that a message would initialise a correlation set with values that are unique wrt the other running sessions in a service would need to be performed at every input that requires to instantiate correlation values. Our type system, instead, performs the required checks statically.

Rule `T-Service` forbids different initialisation methods of a same correlation set, i.e. when two different session branches use the same correlation set they must agree if that correlation set is initialised through local assignments or through the message that started the session. In order to understand why this restriction is necessary, consider the service behaviour

```
[ start1(x) ] { P }
[ start2(y) ] { x = new; Q }
```

where `P` and `Q` are some arbitrary processes. If the behaviour is deployed with aliasing function $\alpha_C = [\texttt{start1} \mapsto (\{x\}, [x \mapsto \epsilon]), \texttt{start2} \mapsto (\{y\}, [y \mapsto \epsilon]),]$ then we could have an undesirable situation where property 2 does not hold. Suppose a session is started through operation `start1`, where `x` would have value v – initialised through (`S-Start`) using the message from the invoker. After that, while the started session is still executing `P`, another session on the same service is started through operation `start2`. The first statement executed by the second session is `x = new`. Because the contract for `new` is to generate *locally*-fresh values `x` could be assigned to the same value v, hence breaking property 2. The constraint introduced by rule `T-Service` for avoiding this situation could be removed by enforcing a stronger contract on the implementation of the `new` primitive, i.e. by making it generating fresh values wrt every other correlation value in the running sessions.