Relational Analysis of Correlation *

Jörg Bauer[1], Flemming Nielson[2], Hanne Riis Nielson[2], and Henrik Pilegaard[2]

[1] Institut für Informatik, Technische Universität München, Germany.
[2] Informatics and Mathematical Modelling, Technical University of Denmark
Kongens Lyngby, Denmark.
joba@model.in.tum.de {nielson,riis,hepi}@imm.dtu.dk

Abstract. In service-oriented computing, correlations are used to determine links between service providers and users. A correlation contains values for some variables received in a communication. Subsequent messages will only be received when they match the values of the correlation. Correlations allow for the implementation of sessions, local shared memory, gradually provided input, or input provided in arbitrary order – thus presenting a challenge to static analysis.
In this work, we present a static analysis in relational form of correlations. It is defined in terms of a fragment of the process calculus COWS that itself builds on the Fusion Calculus. The analysis is implemented and practical experiments allow us to automatically establish properties of the flow of information between services.

1 Introduction

Process calculi have proved their usefulness in describing and analysing distributed systems. It is therefore natural that they are also applied to model and analyse services made available over loosely coupled networks. *Correlation* has been identified as a useful feature to model services and hence appears in emerging service-oriented process calculi [6, 5].

Correlation is an idea that stems from executable business process languages such as BPEL. In the presence of multiple concurrent instances of the same service, messages sent from a user to the provider must be delivered to the correct instance of the service. This is achieved by associating specific, already available data in the messages to maintain a unique reference to a specific service instance. For example, such data may be derived from personal information like a social security number.

Technically, in process calculi, correlation may be realised by the *decoupling of name binding and input actions* in combination with *pattern matching* in input prefixes. Decoupling was first formulated in the Fusion Calculus [14], where inputs are *not* binders. Rather, a *scope construct* $(x)P$ is the only binder binding x in P. The effect of a communication inside P may then be to *fuse* x with another name, e.g., y. The names x and y will then be considered identical. The scope of the subsequent substitution of y for x will then be all of P. This allows

* This work has been partially sponsored by the project SENSORIA, IST-2005-016004.

to model local, shared state, which is essential for correlations. As argued in [3], this decoupling is impossible to encode naturally in π-calculus.

Based on these considerations we have delineated the *Calculus for Web Services*, CoWS. It is mainly a fragment of the Calculus of Orchestration of Web Services (COWS, [6]) developed in the EU project SENSORIA. In contrast to the Fusion Calculus (or D-Fusion developed in [3]), COWS is clearly service-centred and focuses on, among others, correlation. In CoWS, however, we discard some of the technicalities associated with session management.

The goal of this work is not the design of a new language but the development of a relational analysis for correlation. The insights gained here should be easily transferable to any other correlation-based language for services.

Contribution. The study of process calculi is a challenging avenue for the development and application of static analysis. Such analyses often tend to fall between two extremes: rather simple 0-CFA analyses (e.g., [2]) expressing only rudimentary properties, or extremely powerful relational and polyhedral analyses (e.g., [16, 4]). The latter technology seems to be mastered by only a few, and hence might not obtain widespread use, and furthermore seems to require that the calculus in question be presented in a *non-trivial normalised form* in order to aid the analysis thereby making it a major effort to apply the techniques. To be specific, the work of [16] on developing relational and polyhedral analyses for the π-calculus depends heavily on the PhD thesis of [15] that develops the non-trivial normalised form used. We therefore consider it an important achievement of our work that we populate the middle ground between the two extremes – this approach was used successfully in [8] dealing with the π-calculus. Here we apply this approach to develop a relational static analysis for correlation, which is a challenging and highly relevant aspect of process calculi for service-oriented computing. To the best of our knowledge [1], we are the first to develop a static analysis for a descendant of the Fusion Calculus.

Outline. In Section 2, we shall introduce the CoWS calculus illustrating it by giving an example of an accident service. In Section 3, we develop a static analysis in relational form for CoWS. Before we report on our implementation and experimental results, we establish the correctness of our analysis expressed by two major theorems, subject reduction and adequacy. It is noteworthy that our subject reduction result does not claim that analysability is preserved under reduction, but rather that a stronger notion of analysability of all permitted instances is preserved under reduction. This provides a new insight in the development of static analyses of process calculi. Section 5 reports on related work and Section 6 concludes.

2 Calculus for Web Services

Table 1 defines the syntactic domains and the syntax of CoWS, the Calculus for Web Services. We have three syntactic domains, *names*, *variables*, and *labels*.

$s ::=$		Services
	$u \bullet u'!\bar{v}^{\ell}$	(invoke)
$\mid$	g	(input-guarded choice)
$\mid$	$s \mid s$	(parallel composition)
$\mid$	$(n)s$	(name binding box)
$\mid$	$[x]^{\ell}s$	(variable scope)
$\mid$	$*s$	(replication)
$g ::=$		(input-guarded choice)
	$\mathbf{0}$	(nil)
$\mid$	$p \bullet o?\bar{v}^{\ell}.s$	(request processing)
$\mid$	$g + g$	(choice)

Name	**Domain**	**Symbols**
Variables	**Var**	x, y
Names	**Name**	n, m
Partners	**Name**	p, p'
Operations	**Name**	o, o'
	Name $\cup$ **Var**	u, v
Labels	**Lab**	ℓ

Table 1. Syntax of CoWS and its syntactic domains. Service invocation, request processing, as well as the variable scope are decorated with labels, $\ell \in \mathbf{Lab}$. An overline denotes tuples, for instance, $\bar{x}$ denotes tuples of variables.

While the latter merely serve as pointers into the syntax guiding the analysis specification of Section 3, names denote computational values and variables are used to bind names. In the sequel we shall always assume that the services of interest are consistently labelled ensuring that two equally labelled actions are either both inputs – in which case they coincide on partner and operation as well as on the length of the input – or both outputs – in which case they coincide on the length of the output.

The computational entities of CoWS are called *services*. In contrast to, e.g., π-calculus, communication endpoints in CoWS are pairs of *partners* and *operations*, $p \bullet o$. This is employed to model several services available at the same site. Communication endpoints at reception sites are *statically determined*, because a service is supposed to know itself. In contrast, received partner and operation names may be used for subsequent service invocation. Additionally, CoWS has *asynchronous output*, input actions, input-guarded choice, parallel composition, name binding, variable scope declaration, and replication; each with the expected meaning. Names in input prefixes are used for pattern matching.

Variable binding and global scope are the key differences between CoWS and π-like calculi. This is where the similarity to the Fusion Calculus comes into play. Variables are *not* bound at input actions. Rather, the only variable binder is the *scope construct* $[x]^{\ell}s$: If variable x occurs in an input action, at which a name, n, may be received, then the scope of the induced substitution, $[x \mapsto n]$, is the whole of s; not just the continuation of the input action.

2.1 An Accident Service

Our running example is given in Table 2. It describes an arbitrary number of cars (1-3) that have subscribed to the *car accident service* (4-10). Each car is equipped with a GPS device tracking its position. In case an on-board sensor

$*(\mathsf{self})\ *(\mathsf{gps})\ *(\mathsf{sid})$	$\mathsf{p}_s \bullet \mathsf{o}_{alarm}!\langle \mathsf{acc}, \mathsf{self}, \mathsf{sid}\rangle^{1}$	(1)
	$\mid\ (\mathsf{self} \bullet \mathsf{o}_{confirm}?\langle \mathsf{sid}\rangle^{2}.\mathsf{p}_s \bullet \mathsf{o}_{confirm}!\langle \mathsf{ok}, \mathsf{self}, \mathsf{sid}\rangle^{3} +$	(2)
	$\mathsf{self} \bullet \mathsf{o}_{confirm}?\langle \mathsf{sid}\rangle^{4}.\mathsf{p}_s \bullet \mathsf{o}_{confirm}!\langle \mathsf{ko}, \mathsf{gps}, \mathsf{sid}\rangle^{5})$	(3)
$*[x_{info}]^{6}\ [x_{id}]^{7}\ [x_{reply}]^{8}\ [x_{sid}]^{9}$	$\mathsf{p}_s \bullet \mathsf{o}_{alarm}?\langle \mathsf{acc}, x_{id}, x_{sid}\rangle^{10}.$	(4)
	$x_{id} \bullet \mathsf{o}_{confirm}!\langle x_{sid}\rangle^{11}$	(5)
	$\mid\ \mathsf{p}_s \bullet \mathsf{o}_{confirm}?\langle x_{reply}, x_{info}, x_{sid}\rangle^{12}.$	(6)
	$\mathsf{p}_s \bullet \mathsf{o}_{Sos}!\langle x_{reply}, x_{info}, x_{sid}\rangle^{13}$	(7)
	$\mid\ \mathsf{p}_s \bullet \mathsf{o}_{Sos}?\langle \mathsf{ko}, x_{info}, x_{sid}\rangle^{14}.$	(8)
	$\mathsf{p}_{amb} \bullet \mathsf{o}_{Sos}!\langle x_{info}, x_{sid}\rangle^{15}$	(9)
	$\mid\ \mathsf{p}_s \bullet \mathsf{o}_{Sos}?\langle \mathsf{ok}, x_{info}, x_{sid}\rangle^{16}.\mathbf{0}$	(10)

Table 2. An accident service.

detects any abnormal behaviour, the alarm operation of the service centre is invoked by the sending of an accident message containing the identity of the driver and a nonce (1). The latter serves as a unique session identifier preventing malign session interference. The centre processes this request (4) and asks the driver for confirmation (5) in order to exclude false alarms. The driver processes this request (2,3) and either revokes the alarm (2) or confirms it (3). If the alarm is revoked, the driver identity is attached to the (revocation) message. If the alarm is confirmed, the current (GPS) position is attached to the (confirmation) message. In the latter case, one may think of a time-out triggering the confirmation of the accident. In any case, the centre processes the answer (6) and invokes another internal service (7). Depending on the driver's answer this service either calls an external ambulance (8,9), which is informed about the location of the accident, or it terminates due to a false alarm (10).

Note that there may be many cars around that may have a number of false alarms involving different locations. Due to the safety-critical nature of this example, the service centre must be able to handle many of these service calls concurrently without mixing up information from different sessions. In particular, we would like to guarantee that:

1. The ambulance is not called, when an ok was received.
2. Messages sent to the ambulance always contain GPS information. Note, that the validity of this property is not obvious, since x_{info} may be bound to both driver identities and positions at run-time.
3. If several cars employ the accident service at once, then only the positions of the ones confirming the accident are reported to the ambulance.

As we shall see, our analysis is able to provide the desired guarantees.

2.2 Labelled Transition System for CoWS

To define the semantics of CoWS we employ a notion of structural congruence as is typical of process calculi. It is defined as the least congruence that incorporates

$*\mathbf{0} \equiv \mathbf{0}$	[REPL$_1$]
$*s \equiv s \mid *s$	[REPL$_2$]
$(n)\mathbf{0} \equiv \mathbf{0}$	[BINDER$_1$]
$(n)(m)s \equiv (m)(n)s$	[BINDER$_2$]
$s_1 \mid (n)s_2 \equiv (n)(s_1 \mid s_2)$ if $n \notin fn(s_1)$	[BINDER$_3$]
$[x](n)s \equiv (n)[x]s$	[BINDER$_4$]

Table 3. An excerpt of the CoWS structural congruence rules. For a given service expression s, $fn(s)$ denotes the set of free names.

the axioms of Table 3, contains disciplined α-renaming of names[3], and asserts that choice and parallel are associative, commutative, and have $\mathbf{0}$ as neutral element. Note, that we do not allow the extrusion of *variable scopes*. In contrast, [BINDER$_4$], which is a bit unusual, allows the binding boxes of *names* to migrate freely in and out of variable scopes.

Substitutions, σ, are mappings with domain $\mathbf{Var} \to \mathbf{Name}$. The application of a substitution, $[x \mapsto n]$, to a service s is written $s \cdot [x \mapsto n]$ and replaces free occurrences of x in s by n. The disjoint union of two substitutions σ_1 and σ_2 is written $\sigma_1 \uplus \sigma_2$ and the substitution with empty domain is written $\emptyset$.

The CoWS operational semantics as given in Table 4 makes use of the notion of *matching*, where two tuples of names and variables are matched potentially yielding a resulting substitution. The formal definition is as follows:

$$\mathcal{M}(x,n) = x \mapsto n \qquad \mathcal{M}(n,n) = \emptyset \qquad \frac{\mathcal{M}(u_1,v_1) = \sigma_1 \quad \mathcal{M}(\bar{u}_2,\bar{v}_2) = \sigma_2}{\mathcal{M}((u_1,\bar{u}_2),(v_1,\bar{v}_2)) = \sigma_1 \uplus \sigma_2}$$

As we formulate our semantics in terms of a labelled transition system, we shall now define the transition labels α:

$$\alpha ::= (p \bullet o) \lhd \bar{v}^{\ell} \;\mid\; (p \bullet o) \rhd \bar{u}^{\ell} \;\mid\; p \bullet o\lfloor\sigma\rfloor\bar{u}^{\ell_i}\bar{n}^{\ell_o}$$

Transition labels, $(p \bullet o) \lhd \bar{v}^{\ell}$ and $(p \bullet o) \rhd \bar{u}^{\ell}$, result from applying rules for invocation and reception of values, respectively, that is, rules [INV] and [REC] of Table 4. They can engage in a communication ([COM]) resulting in a transition label, $p \bullet o\lfloor\sigma\rfloor\bar{u}^{\ell_i}\bar{n}^{\ell_o}$, where σ is the substitution resulting from the communication. Note, that the only transition labels that denote *observable computation steps* are those of the form $p \bullet o\lfloor\emptyset\rfloor\bar{u}^{\ell_i}\bar{n}^{\ell_o}$. We use $d(\alpha)$ to denote the set of names and variables occurring in α. If $\alpha = p \bullet o\lfloor\sigma\rfloor\bar{u}^{\ell_i}\bar{n}^{\ell_o}$ then $d(\alpha)$ contains only the names and variables in the domain and codomain of σ.

On top level, the semantics of Table 4 is only defined for *closed services*, that is, no free variables may occur when a computation step is applied. Deeper in the inference tree there may of course be free occurrences of variables. The

[3] For disciplined α-renaming, we identify each name with its defining syntactic occurrence, its canonical name, thus partitioning the name space into finitely many equivalence classes. If names are only α-renamed using equivalent names, then canonical names are stable under evaluation.

effect of a substitution of a value for variable x is computed only, when the enclosing scope of x is met ([DEL$_{sub}$]). The substitution transition label is used to propagate the information up to the binding scope. This ensures that the effect of a communication is visible *globally* within a scope. This feature is used to model shared memory of a session instance. Services nested under name bindings or variable scopes can proceed normally unless the enclosing entity is mentioned by the transition label (rules [NAME] and [SCOPE]).

Comparison with COWS. CoWS lacks the orchestration constructs (kill and protect) found in COWS, because we focus on correlation in this work rather than on orchestration. The semantics of CoWS is comparable to the original LTS semantics proposed for COWS. The most notable difference, is in the rule [COM]: Imagine two different communications that both yield a valid match. While our rule picks non-deterministically from the set of choices, the COWS semantics [6] imposes a constraint selecting the match that gives rise to the smallest substitution. In case of equally long substitutions, the choice is random. In [6], this feature is used to distinguish (less instantiated) service definitions from (more instantiated) service instances. However, this particular technicality is not critical for the analysis of correlations presented in the following and therefore it has been omitted from the development. In order to prevent malign session interferences we adapt what is considered good style in communication protocols, where one relies on the use of nonces or shared secrets (or indeed shared keys) to ensure that sessions do not interfere. Note that the role of the nonce in our running example is played by the name sid.

3 A Relational Analysis for CoWS

In this section, we develop a static analysis [9] for CoWS in relational style. That is, for each program label ℓ – service invocation, processing and variable scope – we compute sets of tuples of names to which the variables that are in scope at ℓ may be bound at run-time. The fact that we track *sets* of tuples makes the analysis relational, while an independent attribute analysis would track tuples of sets thus loosing track of which variables are bound at the same time.

Auxiliary Information. We set the stage for the analysis by defining some auxiliary information. First, a *label environment*, L, is defined as a mapping

$$\mathsf{L} : \mathbf{Lab} \to (\mathbf{Var} \times \mathbf{Lab})^*$$

that to each label ℓ associates a sequence $\bar{\chi}$ of pairs of variables and labels that have been introduced before this point in the process; the label indicates the exact scope where the variable was introduced. More formally, we shall take $\mathsf{L} = \mathcal{L}_\epsilon[\![s]\!]$ where $\mathcal{L}_{\bar{\chi}}$ (for $\bar{\chi} \in (\mathbf{Var} \times \mathbf{Lab})^*$) is defined in Table 5. Here we write $\uplus$ for joining two mappings with *disjoint* domains and $[\,]$ for the mapping with empty domain. The notation $\bar{\chi}_1\bar{\chi}_2$ stands for the concatenation of $\bar{\chi}_1$ and $\bar{\chi}_2$. Furthermore, we shall write $x\#\mathsf{L}.\ell$ for the label, at which x at the position ℓ was

$$[\textsc{Inv}] \quad p \bullet o!\bar{n}^{\ell} \xrightarrow{(p\bullet o)\lhd \bar{n}^{\ell}} \mathbf{0} \qquad [\textsc{Rec}] \quad p \bullet o?\bar{u}^{\ell}.s \xrightarrow{(p\bullet o)\rhd \bar{u}^{\ell}} s$$

$$[\textsc{Choice}] \quad \frac{g_1 \xrightarrow{\alpha} s}{g_1 + g_2 \xrightarrow{\alpha} s} \qquad [\textsc{Del}_{sub}] \quad \frac{s \xrightarrow{p\bullet o\lfloor \sigma \uplus [x\mapsto n]\rfloor \bar{u}^{\ell_i}\bar{n}^{\ell_o}} s'}{[x]^{\ell}s \xrightarrow{p\bullet o\lfloor \sigma\rfloor \bar{u}^{\ell_i}\bar{n}^{\ell_o}} s' \cdot [x \mapsto n]}$$

$$[\textsc{Name}] \quad \frac{s \xrightarrow{\alpha} s' \quad n \notin d(\alpha)}{(n)s \xrightarrow{\alpha} (n)s'} \qquad [\textsc{Scope}] \quad \frac{s \xrightarrow{\alpha} s' \quad x \notin d(\alpha)}{[x]s \xrightarrow{\alpha} [x]s'}$$

$$[\textsc{Com}] \quad \frac{s_1 \xrightarrow{(p\bullet o)\rhd \bar{u}^{\ell_i}} s_1' \qquad s_2 \xrightarrow{(p\bullet o)\lhd \bar{n}^{\ell_o}} s_2' \qquad \mathcal{M}(\bar{u},\bar{n}) = \sigma}{s_1 \mid s_2 \xrightarrow{p\bullet o\lfloor \sigma\rfloor \bar{u}^{\ell_i}\bar{n}^{\ell_o}} s_1' \mid s_2'}$$

$$[\textsc{Par}] \quad \frac{s_1 \xrightarrow{\alpha} s_1'}{s_1 \mid s_2 \xrightarrow{\alpha} s_1' \mid s_2} \qquad [\textsc{Cong}] \quad \frac{s \equiv s_1 \quad s_1 \xrightarrow{\alpha} s_2 \quad s_2 \equiv s'}{s \xrightarrow{\alpha} s'}$$

Table 4. CoWS operational semantics.

defined. This is formally defined by reflecting that the most recently introduced definition of x is the rightmost:

$$x\#\langle x_1^{\ell_1}, \ldots, x_k^{\ell_k}\rangle = \ell_t, \text{ if } t = \max\{i \mid x_i = x\}$$

This notion is only defined, when x is really among the x_i, which will always be the case, when we use it.

In the analysis we shall also need a representation of the *flow of control* in the service s of interest. We shall represent this by a *flow mapping* F that to each label ℓ associates the set of labels that will become visible once the action labelled ℓ has been executed; thus

$$\mathsf{F} : \mathbf{Lab} \hookrightarrow \mathcal{P}(\mathbf{Lab})$$

The function $\mathcal{F}$ defined in Table 5 will for each service s define such a mapping together with the set of *visible labels* of the service itself and we shall define $(\mathsf{F}, \mathsf{E}) = \mathcal{F}[\![s]\!]$. When applying F and L to labels, we will mostly write $\mathsf{L}.\ell$ and $\mathsf{F}.\ell$ instead of $\mathsf{L}(\ell)$ and $\mathsf{F}(\ell)$.

Example 1. The label environment computed from the running example in Table 2 is as follows:

$$\mathsf{L}.1 = \mathsf{L}.2 = \mathsf{L}.3 = \mathsf{L}.4 = \mathsf{L}.5 = \mathsf{L}.6 = \epsilon \qquad \mathsf{L}.7 = \langle (x_{info}, 6)\rangle$$
$$\mathsf{L}.8 = \langle (x_{info}, 6), (x_{id}, 7)\rangle \qquad \mathsf{L}.9 = \langle (x_{info}, 6), (x_{id}, 7), (x_{reply}, 8)\rangle$$
$$\mathsf{L}.10 = \mathsf{L}.11 = \cdots = \mathsf{L}.16 = \langle (x_{info}, 6), (x_{id}, 7), (x_{reply}, 8), (x_{sid}, 9)\rangle$$

$$
\begin{array}{rl}
\mathcal{L}_{\bar{\chi}}[\![\mathbf{0}]\!] & = [\,] \\
\mathcal{L}_{\bar{\chi}}[\![u \bullet u'!\bar{v}^{\ell}]\!] & = [\ell \mapsto \bar{\chi}] \\
\mathcal{L}_{\bar{\chi}}[\![p \bullet o?\bar{u}^{\ell}.s]\!] & = \mathcal{L}_{\bar{\chi}}[\![s]\!] \uplus [\ell \mapsto \bar{\chi}] \\
\mathcal{L}_{\bar{\chi}}[\![(n)s]\!] & = \mathcal{L}_{\bar{\chi}}[\![s]\!] \\
\mathcal{L}_{\bar{\chi}}[\![*s]\!] & = \mathcal{L}_{\bar{\chi}}[\![s]\!] \\
\mathcal{L}_{\bar{\chi}}[\![[x]^{\ell}s]\!] & = \mathcal{L}_{\bar{\chi}x^{\ell}}[\![s]\!] \uplus [\ell \mapsto \bar{\chi}] \\
\mathcal{L}_{\bar{\chi}}[\![s_1 \mid s_2]\!] & = \mathcal{L}_{\bar{\chi}}[\![s_1]\!] \uplus \mathcal{L}_{\bar{\chi}}[\![s_2]\!] \\
\mathcal{L}_{\bar{\chi}}[\![s_1 + s_2]\!] & = \mathcal{L}_{\bar{\chi}}[\![s_1]\!] \uplus \mathcal{L}_{\bar{\chi}}[\![s_2]\!]
\end{array}
$$

$$
\begin{array}{rl}
\mathcal{F}[\![\mathbf{0}]\!] & = ([\,], \emptyset) \\
\mathcal{F}[\![u \bullet u'!\bar{v}^{\ell}]\!] & = ([\,], \{\ell\}) \\
\mathcal{F}[\![p \bullet o?\bar{u}^{\ell}.s]\!] & = \text{let } (F, E) = \mathcal{F}[\![s]\!] \\
 & \quad \text{in } (F \uplus [\ell \mapsto E], \{\ell\}) \\
\mathcal{F}[\![(n)s]\!] & = \mathcal{F}[\![s]\!] \\
\mathcal{F}[\![*s]\!] & = \mathcal{F}[\![s]\!] \\
\mathcal{F}[\![[x]^{\ell}s]\!] & = \text{let } (F, E) = \mathcal{F}[\![s]\!] \\
 & \quad \text{in } (F \uplus [\ell \mapsto E], \{\ell\}) \\
\mathcal{F}[\![s_1 \mid s_2]\!] & = \mathcal{F}[\![s_1 + s_2]\!] \\
 & = \text{let } (F_1, E_1) = \mathcal{F}[\![s_1]\!] \\
 & \quad\quad (F_2, E_2) = \mathcal{F}[\![s_2]\!] \\
 & \quad \text{in } (F_1 \uplus F_2, E_1 \cup E_2)
\end{array}
$$

Table 5. Label environment $\mathcal{L}_{\bar{x}}[\![s]\!]$ and flow information $\mathcal{F}[\![s]\!]$

In the sequel, we shall often write pairs like $(x_{info}, 6)$ using superscript as in x^6_{info}. Two examples of flow information are $\mathsf{F}.9 = \{10, 12, 14, 16\}$ and $\mathsf{F}.10 = \{11\}$. Finally, the set of visible labels for the running example is $\{1, 2, 4, 6\}$. □

Analysis Domain. The *abstract environments* $\hat{\mathsf{R}}$ of the analysis will, given a label ℓ, return a set of sequences. Each element of such a sequence can either be a name or the *undefined* symbol $\perp$. The latter denotes cases, where a variable has not been assigned a value yet. This helps to track gradually provided inputs. The length of these sequences will equal that of $\mathsf{L}.\ell$. We will thus determine the potential values of the variables at the point determined by ℓ and take:

$$\hat{\mathsf{R}} : \mathbf{Lab} \to \mathcal{P}(\mathbf{Name}^*_{\perp})$$

where $\mathbf{Name}_{\perp} = \mathbf{Name} \cup \{\perp\}$ and $\mathbf{Name}^*_{\perp}$ denotes tuples of elements from $\mathbf{Name}_{\perp}$.[4] We shall use w to denote elements of $\mathbf{Name}_{\perp}$. If $\hat{\mathsf{R}}.\ell = \emptyset$ it means that the program point ℓ is not reachable; if $\hat{\mathsf{R}}.\ell = \{\epsilon\}$ then also $\mathsf{L}.\ell = \epsilon$ and it means that no variable has been introduced at that program point.

We use the following auxiliary function to determine the potential values of a variable x at the label ℓ: $\mathit{\Pi}_{x@\mathsf{L}.\ell}(\bar{w})$, where $\bar{w}$ has the same length as $\mathsf{L}.\ell$ and x occurs in $\mathsf{L}.\ell$. In the cases where more than one occurrence of x occurs in $\mathsf{L}.\ell$ we always select the rightmost – corresponding to the most recently declared one. The function is defined by

$$\mathit{\Pi}_{x@\mathsf{L}.\ell}(\bar{w}) = w_t$$

where $\bar{w} = \langle w_1, \ldots, w_k \rangle$, $\mathsf{L}.\ell = \langle x_1^{\ell_1}, \ldots, x_k^{\ell_k} \rangle$, and $t = \max\{i \mid x_i = x\}$. Note, that the result may be $\perp$. The operation is trivially extended to names, n:

$$\mathit{\Pi}_{n@\mathsf{L}.\ell}(\bar{w}) = n$$

[4] Again, all the names tracked in the analysis are canonical, that is they are identified with their syntactic occurrence.

$$[\text{RNIL}]\ \ \hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} \mathbf{0} \qquad [\text{RREP}]\ \ \frac{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s}{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} *s} \qquad [\text{RNAME}]\ \ \frac{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s}{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} (n)s}$$

$$[\text{RPAR}]\ \ \frac{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s_1 \quad \hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s_2}{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s_1 \mid s_2} \qquad [\text{RCHOICE}]\ \ \frac{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s_1 \quad \hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s_2}{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s_1 + s_2}$$

$$[\text{RINV}]\ \hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} u \bullet u'!\bar{v}^{\ell} \quad \textbf{if}\ \Pi_{uu'\bar{v}@\mathsf{L}.\ell}(\hat{\mathsf{R}}.\ell) \cap \mathbf{Name}^* \subseteq \hat{\mathsf{K}}$$

$$[\text{RSCOPE}]\ \frac{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s}{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} [x]^{\ell} s} \quad \textbf{if}\ \forall \ell' \in \mathsf{F}.\ell : \hat{\mathsf{R}}.\ell \times \{\bot\} \subseteq \hat{\mathsf{R}}.\ell'$$

$$[\text{RREC}]\ \frac{X \neq \emptyset\ \Rightarrow\ \hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s}{\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} p \bullet o?\bar{u}^{\ell}.s} \quad \textbf{if} \begin{array}{l} \forall \ell' \in \mathsf{F}.\ell : X \subseteq \hat{\mathsf{R}}.\ell' \\ \forall x \in \{\bar{u}\}\ \forall \ell_x \in (\mathsf{F}.(x\#\mathsf{L}.\ell)) : Y_x \subseteq \hat{\mathsf{R}}.\ell_x \end{array}$$

$$\begin{array}{ll} \textbf{where} & X = \{\bar{w} \in \hat{\mathsf{R}}.\ell \mid \Pi_{po\bar{u}@\mathsf{L}.\ell}(\bar{w}) \in \hat{\mathsf{K}}\} \\ \textbf{and} & Y_x = \{\bar{w}n \in \mathbf{Name}^*_{\bot} \mid \bar{w} \in \hat{\mathsf{R}}.(x\#\mathsf{L}.\ell) \wedge \\ & \qquad\qquad \exists \bar{w}' \in \mathbf{Name}^*_{\bot} : \Pi_{po\bar{u}@\mathsf{L}.\ell}(\bar{w}n\bar{w}') \in \hat{\mathsf{K}}\ \} \end{array}$$

Table 6. Specification of the analysis judgement $\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s$.

Also, it is extended to sequences, $\bar{u}$, of variables and names, as in $\Pi_{\bar{u}@\mathsf{L}.\ell}(\bar{w})$, and to sets, R, of such sequences, i.e., $\Pi_{\bar{u}@\mathsf{L}.\ell}(R)$.

The abstract communication cache

$$\hat{\mathsf{K}} \subseteq \mathbf{Name} \times \mathbf{Name} \times \mathbf{Name}^*$$

records the tuples of names that potentially are communicated over the channels. Elements of the domain are triples representing a partner name, an operation name, and the tuple of names communicated. In $\hat{\mathsf{K}}$, we keep track of names only, there is no $\bot$ involved.

Analysis Specification. The *judgements* of the analysis have the form

$$\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s$$

where L, F, $\hat{\mathsf{R}}$, and $\hat{\mathsf{K}}$ are as above. Intuitively, the judgements defined in Table 6 determine whether a given pair $(\hat{\mathsf{R}}, \hat{\mathsf{K}})$ is a valid analysis result.

The first five rules are simple recursive cases. The rule [RINV] deals with invocations. It looks up the available bindings of the involved variables in $\hat{\mathsf{R}}$ and records the resulting tuples in the abstract communication cache, $\hat{\mathsf{K}}$. Since the lookup may yield undefined values, we need to intersect with $\mathbf{Name}^*$.

The rule [RSCOPE] takes care of scope definitions. It extends whatever variable bindings that are available at a scope definition, ℓ, with a single $\bot$ and passes the information on to the program points following ℓ, reflecting the fact that the newly introduced variable is not yet bound.

Finally, the rule [RREC] describes two different flows. First, it ensures that all possible bindings at ℓ that may lead to a tuple that might be communicated will indeed flow to the subsequent visible labels (expressed in the set X); only if this set is non-empty is it possible to perform the input and hence the test $X \neq \emptyset$ expresses a reachability condition for the continuation s. Second, for each variable x of the input pattern, we record (in the set Y_x) which names may be bound to x by any communication that matches this input pattern. This information flows to the labels just after the scope at which x was introduced (denoted by $\mathsf{F}.(x\#\mathsf{L}.\ell)$).

In addition to the satisfaction of the analysis judgement, we require some initialisation information, stating that ϵ is available at all globally visible labels:

Definition 1. *Let s be a service, $(\mathsf{F},\mathsf{E}) = \mathcal{F}[\![s]\!]$ its flow information, and $\mathsf{L} = \mathcal{L}_\epsilon[\![s]\!]$ its label environment. A pair $(\hat{\mathsf{R}},\hat{\mathsf{K}})$ is an* acceptable analysis estimate for *s, if and only if $\hat{\mathsf{R}},\hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s$ and $\epsilon \in \hat{\mathsf{R}}.\ell$ for all $\ell \in \mathsf{E}$.*

Example 2. An acceptable analysis estimate – and in fact the least, that is, most precise one – of the running example of Table 2 comprises:

$$\begin{aligned}\hat{\mathsf{K}} = \{\ &\langle \mathsf{p}_s, \mathsf{o}_{alarm}, \mathsf{acc}, \mathsf{self}, \mathsf{sid}\rangle, \langle \mathsf{p}_s, \mathsf{o}_{confirm}, \mathsf{ok}, \mathsf{self}, \mathsf{sid}\rangle,\\ &\langle \mathsf{p}_s, \mathsf{o}_{confirm}, \mathsf{ko}, \mathsf{gps}, \mathsf{sid}\rangle, \langle \mathsf{self}, \mathsf{o}_{confirm}, \mathsf{sid}\rangle, \langle \mathsf{p}_s, \mathsf{o}_{Sos}, \mathsf{ok}, \mathsf{self}, \mathsf{sid}\rangle,\\ &\langle \mathsf{p}_s, \mathsf{o}_{Sos}, \mathsf{ko}, \mathsf{gps}, \mathsf{sid}\rangle, \langle \mathsf{p}_{amb}, \mathsf{o}_{Sos}, \mathsf{gps}, \mathsf{sid}\rangle\ \}\end{aligned}$$

Regarding the abstract environments, we state only $\hat{\mathsf{R}}.13$ and $\hat{\mathsf{R}}.15$ explicitly, because they are most relevant with respect to the properties we are interested in. Recall that both $\mathsf{L}.13$ and $\mathsf{L}.15$ amount to $\langle (x_{info},6),(x_{id},7),(x_{reply},8),(x_{sid},9)\rangle$.

$$\begin{aligned}\hat{\mathsf{R}}.13 &= \{\langle \mathsf{self},\mathsf{self},\mathsf{ok},\mathsf{sid}\rangle, \langle \mathsf{self},\bot,\mathsf{ok},\mathsf{sid}\rangle, \langle \mathsf{gps},\mathsf{self},\mathsf{ko},\mathsf{sid}\rangle, \langle \mathsf{gps},\bot,\mathsf{ko},\mathsf{sid}\rangle\}\\ \hat{\mathsf{R}}.15 &= \{\langle \mathsf{gps},\mathsf{self},\mathsf{ko},\mathsf{sid}\rangle, \langle \mathsf{gps},\mathsf{self},\bot,\mathsf{sid}\rangle, \langle \mathsf{gps},\bot,\mathsf{ko},\mathsf{sid}\rangle, \langle \mathsf{gps},\bot,\bot,\mathsf{sid}\rangle\}\end{aligned}$$

Reconsider the three properties stated in Section 2.1. Property 1 states that the ambulance is not called, when an ok was received. The ambulance is called at label 15. In the analysis result, we can see that the value of x_{reply} (the third position in the tuples) cannot be ok proving property 1. Note that at label 13, ok may still occur.

Property 2 requires that an ambulance is only called with location information. This is shown by inspecting $\hat{\mathsf{K}}$, which over-approximates all messages sent. The only message involving p_{amb} in $\hat{\mathsf{K}}$ contains gps only, thereby proving property 2.

Property 3 is not so easy to show for the analysis as is. We are able to establish it, if we unfold the definition of cars a finite number of times, though. The analysis will then show, that malign interferences are prevented by the session id sid. The formal result justifying our reasoning about properties in this example is stated in Theorem 2 below. □

4 Properties of the Analysis

In this section, we shall establish the formal correctness of our analysis. We start by defining a *correctness predicate*, which states the analysability of *all permitted substitutions* of an analysable service, and show that its validity is preserved under observable computation steps. This constitutes our subject reduction result and is formalised in Theorem 1. It is shown in Appendix A that mere analysability is not preserved under reduction. Theorem 2 states that all actually sent messages and all potential variable bindings are correctly recorded in the analysis. We conclude this section by reporting on experiments using the implementation of our analysis. The feasibility of this implementation relies on Theorem 3 stating the Moore family property of the set of all acceptable analyses; hence guaranteeing the existence of least (most precise) solutions.

4.1 Correctness

In the following we assume an arbitrary but fixed given program $s_\star$, as well as its flow information $\mathsf{F} = \mathcal{F}[\![s_\star]\!]$, its label and its visible labels $(\mathsf{L}, \mathsf{E}) = \mathcal{L}_\epsilon[\![s_\star]\!]$. Moreover, we define the notion $\mathcal{E}(s)$ of the *exposed actions* of a service s, which is a set of input and output prefixes, such that $\mathcal{E}(u \bullet u'!\bar{v}^\ell) = \{u \bullet u'!\bar{v}^\ell\}$, $\mathcal{E}(p \bullet o?\bar{v}^\ell.s) = \{p \bullet o?\bar{v}^\ell.s\}$, $\mathcal{E}(*s) = \mathcal{E}((n)s) = \mathcal{E}([x]s) = \mathcal{E}(s)$, $\mathcal{E}(\mathbf{0}) = \emptyset$, $\mathcal{E}(s_1 \mid s_2) = \mathcal{E}(s_1 + s_2) = \mathcal{E}(s_1) \cup \mathcal{E}(s_2)$. Finally, we define an extended version of substitution. Let $s = p \bullet o?\bar{v}^\ell.s'$ or $s = u \bullet u'!\bar{u}^\ell$ and let $\bar{w} \in \hat{\mathsf{R}}.\ell$ where $\mathsf{L}.\ell = \langle x_1, \ldots, x_k \rangle$ and $\bar{w} = \langle w_1, \ldots, w_k \rangle$. Then we define

$$s[\bar{w}/\mathsf{L}.\ell] = (\ldots(s \cdot [x_k \mapsto w_k]) \cdot \ldots) \cdot [x_1 \mapsto w_1]$$

where $s \cdot [x \mapsto \bot] = s$.

Intuitively, the correctness predicated defined in Definition 2 holds of a service s obtained by semantic reduction from $s_\star$, if all exposed actions s' of s have a counterpart s'' in $s_\star$, such that s'' is correctly analysed and such that s'' is congruent to s' when instantiating it with information computed by the analysis. In other words, the correctness predicate describes the analysability of permitted substitution instances.

Definition 2 (Correctness Predicate). *A service s satisfies the correctness predicate with respect to $s_\star$, written $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s$ if and only if (1-4) hold.*

1. $\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s_\star$
2. $\forall \ell \in \mathsf{E} : \epsilon \in \hat{\mathsf{R}}.\ell$
3. *For all $p \bullet o?\bar{u}^\ell.s' \in \mathcal{E}(s)$ there exists a subexpression $p \bullet o?\bar{v}^\ell.s''$ of $s_\star$ s.t.*
   - $\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} p \bullet o?\bar{v}^\ell.s''$ *and*
   - $\exists \bar{w} \in \hat{\mathsf{R}}.\ell : p \bullet o?\bar{u}^\ell.s' \equiv p \bullet o?\bar{v}^\ell.s''[\bar{w}/\mathsf{L}.\ell]$
4. *For all $v \bullet v'!\bar{v}^\ell \in \mathcal{E}(s)$ there exists a subexpression $u \bullet u'!\bar{u}^\ell$ of $s_\star$ s.t.*
   - $\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} u \bullet u'!\bar{u}^\ell$ *and*
   - $\exists \bar{w} \in \hat{\mathsf{R}}.\ell : v \bullet v'!\bar{v}^\ell \equiv u \bullet u'!\bar{u}^\ell[\bar{w}/\mathsf{L}.\ell]$

The following lemma, which shall be used in the proof of Theorem 1, states some obvious compositionality properties of the correctness predicate. The proof is straightforward from Definition 2.

Lemma 1 (Compositionality). *Let s, s_1, s_2 be services, x a variable and n a name. It holds:*

- $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_1 \mid s_2$ *if and only if* $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_1$ *and* $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_2$.
- $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_1 + s_2$ *if and only if* $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_1$ *and* $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_2$.
- $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} (n)s$ *if and only if* $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s$.
- $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} [x]^\ell s$ *if and only if* $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s$.

Moreover, if $s_1 \equiv s_2$ then $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_1$ if and only if $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_2$.

We are now able to state our subject reduction result. Note that the correctness predicate is only preserved on top-level, that is, when talking about observable computation steps. A computation step is observable, when the substitution of the transition label is empty, that is, when all substitutions induced by a communication have happened.

Theorem 1 (Subject Reduction). *Let s_1 and s_2 be services. If $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_1$ and $s_1 \xrightarrow{p \bullet o \lfloor \emptyset \rfloor \bar{u}^{\ell_i} \bar{n}^{\ell_o}} s_2$ then $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star} s_2$.*

The somewhat technical proof of the theorem is presented in Appendix B. In fact, it requires an even stronger induction hypothesis than provided by the correctness predicate. This is because we have to deal with all possible transition labels, in particular with transition labels carrying non-empty substitutions. The stronger induction hypothesis then makes a connection between these substitutions and the analysis result.

The following theorem constitutes the adequacy of our analysis. It states that every communication triggering an observed substitution and the substitution itself are in fact recorded in the analysis information. The proof follows directly from Theorem 1 and its proof in Appendix B.

Theorem 2 (Adequacy). *If $(\hat{\mathsf{R}}, \hat{\mathsf{K}})$ is an acceptable analysis of $s_\star$ and if $s_\star \rightarrow^* s \xrightarrow{p \bullet o \lfloor \emptyset \rfloor \bar{u}^{\ell_o} \bar{n}^{\ell_i}} s'$ then $\langle po\bar{n} \rangle \in \hat{\mathsf{K}}$ and $\bar{n} \in \Pi_{\bar{u}@\mathsf{L}.\ell_i}(\hat{\mathsf{R}}.\ell_i)$.*

4.2 Implementation

The basis of our implementation is the following Moore family result ensuring both the existence and the uniqueness of a least acceptable analysis result.

Theorem 3 (Moore Family). *For any service s the set of acceptable analysis estimates under $\vdash_{\mathsf{L},\mathsf{F}}$ constitutes a Moore family, i.e.,*

$$\forall A \subseteq \{\hat{\mathsf{R}}, \hat{\mathsf{K}} \mid \hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s\} : \sqcap A \in \{\hat{\mathsf{R}}, \hat{\mathsf{K}} \mid \hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s\}.$$

Proof. As the analysis specification is syntax directed the result follows by straightforward structural induction on s. □

If we, by a change of perspective, view the analysis specification as logical formulas and acceptable results as models of these formulas, then the Moore family result turns into a model intersection property ensuring a least model corresponding to the least acceptable analysis result.

We have implemented a fully functional prototype in Standard ML generating clauses that lie within the Alternation-free Least Fixed Point (ALFP) fragment of first order logic. Least models of such formulas always exist and can be computed efficiently by, e.g., the Succinct Solver [12, 10].

Example 3. The input communication of line (6) of Table 2 gives rise to the following clause:

$$\begin{aligned}&\forall x_{info}, x_{id}, x_{reply}, x_{sid} :\\ &\qquad \hat{\mathsf{R}}.12(x_{info}, x_{id}, x_{reply}, x_{sid}) \wedge \hat{\mathsf{K}}(\mathsf{p}_s, \mathsf{o}_{confirm}, x_{reply}, x_{info}, x_{sid}) \Rightarrow\\ &\qquad\qquad [\ \hat{\mathsf{R}}.13(x_{info}, x_{id}, x_{reply}, x_{sid}) \wedge \phi \wedge \psi\ ]\end{aligned}$$

This specifies the flow into $\hat{\mathsf{R}}.13$: All variable bindings available at $\hat{\mathsf{R}}.12$ leading to a tuple that may be communicated flow to the continuation at line (7). This corresponds to the X set in rule [RRec]. Formula ϕ in the clause above corresponds to the Y_x set and is left out for brevity. Formula ψ in the clause above corresponds to line (7) of the example representing the analysis of an output according to [RInv] and specifying a flow into $\hat{\mathsf{K}}$

$$\begin{aligned}&\psi = \forall x_{info}, x_{id}, x_{reply}, x_{sid} :\\ &\qquad \hat{\mathsf{R}}.13(x_{info}, x_{id}, x_{reply}, x_{sid}) \wedge x_{sid} \neq \bot \wedge x_{info} \neq \bot \wedge x_{reply} \neq \bot \Rightarrow\\ &\qquad\qquad \hat{\mathsf{K}}(\mathsf{p}_s, \mathsf{o}_{Sos}, x_{reply}, x_{info}, x_{sid})\end{aligned}$$

□

Complexity. Three quantities determine the complexity of solving the derived ALFP clause, the number n of names used in the program the maximal nesting depth of variables, bounded by $m = \max_{\ell \in \mathbf{Lab}} |\mathsf{L}.\ell|$, and the maximal length of any sent message, bounded by $k = \max_{u \bullet u' ! \bar{u} \in s} |\bar{u}|$. The size of the logical universe is bounded by n, while m and n decide the maximal arity of relations. Furthermore, the maximal nesting depth is decided by m, which, by Proposition 1 of [11], results in a complexity bound of $\mathcal{O}(n^{3+k+m})$. This is exponential in the worst case, which is only realised, however, by service specifications, s, where the number of sequenced inputs (m) and/or the arity of sent messages (k) are/is linear in the size (n) of s. For most realistic CoWS specifications the complexity will be polynomial, and for the accident service the solution was found in less than a second.

5 Related Work

The separation of the notions of scope and binding occurrence of a variable is originally due to Parrow and Victor, who used it to model a notion of locally shared memory in the Fusion Calculus [14]. In contrast to CoWS, there is only one syntactic category, names, and input and output are completely symmetric. Symmetric means that input and output designation can be exchanged while still yielding the same fusion. Therefore, the authors of [14] suggest to use the terminology action and co-action instead.

In [3], the calculus D-Fusion is proposed. It extends the Fusion Calculus with another binder similar to the restriction of π-calculus to obtain more expressiveness. It retains the symmetry of actions, in fact, it does not even distinguish input and output at all. Apart from the symmetry of input and output, the D-Fusion calculus introduced in [3] is quite close to CoWS. However, we prefer to consider CoWS as a subset of the COWS [6] calculus, because the latter is more focused on correlations and their use in service-oriented computing.

COWS retains a variant of separation of input and name binding in order to faithfully model correlation. In contrast to Fusion and D-Fusion, inputs and outputs are clearly distinguished, in particular by using two syntactic categories, names and variables, where only substitutions of names for variables are possible. COWS also features pattern matching in input prefixes facilitating correlations. While we, too, embrace these concepts we discard some of the technicalities associated with session management and do not consider the fault and compensation facilities provided by COWS.

To the best of our knowledge [1] the present static analysis is the first static analysis, which is not a type system, developed for the Fusion Calculus or its descendants. Also, it seems to be the first static analysis of correlation. Our analysis is relational in form. This form has previously been investigated in the simpler context of the π-calculus [8] equipped with a reaction style semantics. The fact that the present development retains the simplicity of the former, even in the context of a more complicated calculus equipped with a labelled structural operational semantics, testifies to the flexibility of the Flow Logic specification style [13]. In contrast, previous relational approaches [16, 4], fashioned within the framework of Abstract Interpretation, have relied on highly customised syntax and semantics and are not easily extended beyond the original context of the π-calculus.

In the context of COWS, Lapadula et. al. use Type Systems in order to enforce a set of distribution policy annotations [7]. For this approach to work the user has to annotate each piece of data with a region of maximal dissemination (a set of service principals). Static inference combined with a typed semantics, performing appropriate run-time checks, then ensures that the policy is never violated. In contrast, our approach relies neither on user-provided annotations, nor dynamic type-checking. Rather it is fully static and automatically computes a very precise estimate of the data-sets that may reach every single program point. The specified information is very general; hence a region based policy can easily

be tested against the computed result. In [7] the complexity of the type system is not discussed, but the general tendency is that checking is polynomial, whereas inference is exponential. In the case of the Succinct Solver, however, there is no complexity gap between checking and inference [12] – both are polynomial for non-pathological specifications.

6 Conclusion

In this paper we have delineated the process calculus CoWS for implementing correlation based services in order to focus on the essentials of correlation, e.g., separation of binding and input and pattern matching. Therefore, we expect that our analysis of CoWS transfers easily to any other correlation based process calculus. CoWS is formulated in a form resembling COWS but lacking its orchestration constructs (kill and protect). In future extensions of our work we aim at adding these missing primitives.

We then developed a relational analysis for the CoWS process calculus and showed its usefulness for ensuring that service invocations do not interfere in malign ways. We have based our work on a recent relational analysis developed for the π-calculus [8] thereby supporting the claim that the Flow Logic framework facilitates transferring analysis insights between languages (being programming languages or process calculi). While more powerful approaches to relational analysis exist, in particular the work of polyhedral analysis of certain π-processes [16, 4], they are substantially harder to transfer to other language because they rely on a special "normal form" for processes to have been established a priori (and in the case of the π-calculus in the PhD-thesis of [15]).

Despite the guidance offered by the Flow Logic framework and the relational analysis developed for the π-calculus in [8] correlations or, technically, the separation of scope from binding have presented profound obstacles that we have managed to solve. A key ingredient is the use of $\perp$ to denote the "presence" of a variable that has not yet received its value; this technique is being used in rather deep ways to ensure the semantic correctness of the static analysis. The correctness result follows the approach first pioneered in [8] in making use of a subject reduction result where "analysability" is not preserved under reduction whereas the more complex notion of "analysability of all permitted substitution instances" is.

References

1. Personal communication with B. Victor, October 2007.
2. C. Bodei, P. Degano, F. Nielson, and H. R. Nielson. Static analysis for the π-calculus with applications to security. *Information and Computation*, 168:68–92, 2001.
3. M. Boreale, M. G. Buscemi, and U. Montanari. D-fusion: A distinctive fusion calculus. In W.-N. Chin, editor, *APLAS*, volume 3302 of *Lecture Notes in Computer Science*, pages 296–310. Springer, 2004.

4. J. Feret. Dependency analysis of mobile systems. In D. L. Métayer, editor, *ESOP*, volume 2305 of *Lecture Notes in Computer Science*, pages 314–330. Springer, 2002.
5. C. Guidi, R. Lucchi, R. Gorrieri, N. Busi, and G. Zavattaro. Sock: A calculus for service oriented computing. In A. Dan and W. Lamersdorf, editors, *ICSOC*, volume 4294 of *Lecture Notes in Computer Science*, pages 327–338. Springer, 2006.
6. A. Lapadula, R. Pugliese, and F. Tiezzi. A calculus for orchestration of web services. In R. De Nicola, editor, *Proc. of 16th European Symposium on Programming (ESOP'07)*, Lecture Notes in Computer Science. Springer, 2007.
7. A. Lapadula, R. Pugliese, and F. Tiezzi. Regulating data exchange in service oriented applications. In *Proc. of IPM International Symposium on Fundamentals of Software Engineering (FSEN'07)*, volume 4767 of *Lecture Notes in Computer Science*, pages 223–239. Springer, 2007.
8. F. Nielson, H. R. Nielson, J. Bauer, C. R. Nielsen, and H. Pilegaard. Relational analysis for delivery of services. In *Trustworthy Global Computing*, 2007. To appear.
9. F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999.
10. F. Nielson, H. R. Nielson, H. Sun, M. Buchholtz, R. R. Hansen, H. Pilegaard, and H. Seidl. The succinct solver suite. In *TACAS*, pages 251–265, 2004.
11. F. Nielson and H. Seidl. Control-flow analysis in cubic time. In *Proc. ESOP'01*, number 2028 in Lecture Notes in Computer Science, pages 252–268. Springer, 2001.
12. F. Nielson, H. Seidl, and H. R. Nielson. A succinct solver for ALFP. *Nord. J. Comput.*, 9(4):335–372, 2002.
13. H. R. Nielson and F. Nielson. Flow Logic: a multi-paradigmatic approach to static analysis. In T. Mogensen, D. Schmidt, and I. H. Sudburough, editors, *The Essence of Computation: Complexity, Analysis, Transformation. Essays dedicated to Neil D. Jones*, volume 2566 of *Lecture Notes in Computer Science*, pages 223–244. Springer Verlag, 2002.
14. J. Parrow and B. Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *LICS*, pages 176–185, 1998.
15. D. N. Turner. *The Polymorphic Pi-calulus: Theory and Implementation*. PhD thesis, University of Edinburgh, 1996.
16. A. Venet. Automatic determination of communication topologies in mobile systems. In G. Levi, editor, *SAS*, volume 1503 of *Lecture Notes in Computer Science*, pages 152–167. Springer, 1998.

A Subject Reduction Revisited

We have established a substitution based subject reduction result in Theorem 1 based upon a correctness predicate relating correct analyses results to substitution instances of the programme originally analysed. Now, we shall present a counterexample illustrating that general subject reduction with respect to correct analyses fails. Consider the following service and a computation step:

$$\begin{array}{ll}[x]^1 & p \bullet o?x^2.\mathbf{0} \\ & \mid p \bullet o!\mathsf{a}^3 \\ & \mid p \bullet o?x^4.p \bullet o!\langle\rangle^5\end{array} \xrightarrow{p \bullet o \lfloor \emptyset \rfloor x^2 \mathsf{a}^3} p \bullet o?\mathsf{a}^4.p \bullet o!\langle\rangle^5$$

It is obvious that a valid analysis result of the left hand side may comprise $\hat{\mathsf{R}}.4 = \{\mathsf{a}, \bot\}$ and $\hat{\mathsf{R}}.5 = \{\mathsf{a}\}$, because the X in [Req] filters $\bot$ away. However, analysing

the right-hand side in isolation requires $X = \{\mathsf{a}, \bot\} \subseteq \hat{\mathsf{R}}.5$ which is clearly not satisfied by the result given. The filter does not work in this case constituting an example of standard subject reduction not holding. A key insight is that correct analyses are not preserved whenever we have a match involving more than constants, e.g. variables. More generally speaking, the problem occurs when we have a static analysis with localised environments, where we incorporate the results of matchings into these environments. This observation becomes apparent in [8], too.

B Proof of Theorem 1

In order for the proof to work, we need to define a stronger induction hypothesis in form of a correctness predicate parameterised by a substitution σ. Definition 2 remains unchanged except for the fact that we require the substitution to be compatible with the element $\bar{w}$ of the analysis result that we use to create a substitution instance with. We shall write $\overline{\{x_1, \ldots, x_k\}}$ to denote the vector $\langle x_1 \ldots x_k \rangle$ (the concrete order is irrelevant) and $var(\bar{u})$ to denote the vector that remains of $\bar{u}$ after removing all non-variables.

Definition 3 (Parameterised Correctness Predicate). *A service s satisfies the parameterised correctness predicate with respect to $s_\star$ and σ, written $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s$ if and only if (1-4) hold.*

1. $\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} s_\star$
2. $\forall \ell \in \mathsf{E} : \epsilon \in \hat{\mathsf{R}}.\ell$
3. *For all $p \bullet o?\bar{u}^\ell.s' \in \mathcal{E}(s)$ there exists a subexpression $p \bullet o?\bar{v}^\ell.s''$ of $s_\star$ s.t.*
   - $\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} p \bullet o?\bar{v}^\ell.s''$ *and*
   - $\exists \bar{w} \in \hat{\mathsf{R}}.\ell : p \bullet o?\bar{u}^\ell.s' \equiv p \bullet o?\bar{v}^\ell.s''[\bar{w}/\mathsf{L}.\ell]$ *and* $\mathit{\Pi}_{\overline{dom(\sigma)}@\mathsf{L}.\ell}(\bar{w}) = \sigma(\overline{dom(\sigma)})$.
4. *For all $v \bullet v'!\bar{v}^\ell \in \mathcal{E}(s)$ there exists a subexpression $u \bullet u'!\bar{u}^\ell$ of $s_\star$ s.t.*
   - $\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} u \bullet u'!\bar{u}^\ell$ *and*
   - $\exists \bar{w} \in \hat{\mathsf{R}}.\ell : v \bullet v'!\bar{v}^\ell \equiv u \bullet u'!\bar{u}^\ell[\bar{w}/\mathsf{L}.\ell]$ *and* $\mathit{\Pi}_{\overline{dom(\sigma)}@\mathsf{L}.\ell}(\bar{w}) = \sigma(\overline{dom(\sigma)})$.

Notice that Definition 3 and Definition 2 coincide for empty substitutions. Therefore, the proof of Theorem 1 follows directly from case 3 of Lemma 2 below, when we choose $\sigma = \emptyset$.

Lemma 2. *Let s_1 and s_2 be services. The following statements hold.*

1. *For all $\sigma : \mathbf{Var} \to \mathbf{Name}$ holds: If $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_1$ and $s_1 \xrightarrow{(p\bullet o) \lhd \bar{n}^\ell} s_2$ then $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_2$ and $\langle po\bar{n} \rangle \in \hat{\mathsf{K}}$.*
2. *For all $\sigma : \mathbf{Var} \to \mathbf{Name}$ such that $dom(\sigma) = \{x \mid x \in \{\bar{u}\}\}$ holds: If $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_1$ and $s_1 \xrightarrow{(p\bullet o) \rhd \bar{u}^\ell} s_2$ and $po\bar{u} \cdot \sigma \in \hat{\mathsf{K}}$ then $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_2$.*

3. If $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_1$ *and* $s_1 \xrightarrow{p \bullet o \lfloor \sigma \rfloor \bar{u}^{\ell_i} \bar{n}^{\ell_o}} s_2$ *then* $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_2$.

When proving Lemma 2 we make use of the fact, that Lemma 1 holds for the parameterised correctness predicate and all substitutions. All three parts of Lemma 2 are proven by induction on the inference tree used to establish the semantic reduction. We start with the base case of case 1, that is, rule [Inv]. Let σ be an arbitrary substitution and let

$$p \bullet o!\bar{n}^{\ell} \xrightarrow{(p \bullet o) \lhd \bar{n}^{\ell}} \mathbf{0}$$

such that $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} p \bullet o!\bar{n}^{\ell}$. It is obvious that $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} \mathbf{0}$. We know that there exists $u \bullet u'!\bar{u}^{\ell}$ in $s_\star$ and $\bar{w} \in \hat{\mathsf{R}}.\ell$ such that $uu'\bar{u}[\bar{w}/\mathsf{L}.\ell] = po\bar{n}$. Hence by rule [RInv], we obtain $po\bar{n} \in \hat{\mathsf{K}}$. The inductive cases follow easily by (the variation of) Lemma 1.

When we prove part 2 of Lemma 2, it suffices to concentrate on the base case, that is, rule [Rec]. So assume

$$p \bullet o?\bar{u}^{\ell}.s \xrightarrow{(p \bullet o) \rhd \bar{u}^{\ell}} s$$

and let σ be an appropriate substitution such that $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} p \bullet o?\bar{u}^{\ell}.s$. We know that there exists a $p \bullet o?v^{\ell}.s'$ in $s_\star$ and $\bar{w} \in \hat{\mathsf{R}}.\ell$ such that

$$\hat{\mathsf{R}}, \hat{\mathsf{K}} \vdash_{\mathsf{L},\mathsf{F}} p \bullet o?v^{\ell}.s' \tag{1}$$

$$p \bullet o?u^{\ell}.s \equiv p \bullet o?v^{\ell}.s'[\bar{w}/\mathsf{L}.\ell] \tag{2}$$

$$\mathit{\Pi}_{\overline{dom(\sigma)}@\mathsf{L}.\ell}(\bar{w}) = \sigma(\overline{dom(\sigma)}) \tag{3}$$

Effectively, according to analysis rule [RRec], it suffices to show, that $\bar{w} \in X$ (referring to the X in [RRec]), that is, $\mathit{\Pi}_{po\bar{v}@\mathsf{L}.\ell}(\bar{w}) \in \hat{\mathsf{K}}$. Since we know $po\bar{u} \cdot \sigma \in \hat{\mathsf{K}}$ by the assumption of part 2 of Lemma 2, and the combination of (2) and (3) yields $(p \bullet o?u^{\ell}.s) \cdot \sigma \equiv p \bullet o?v^{\ell}.s'[\bar{w}/\mathsf{L}.\ell]$ we can conclude this requirement and thus the proof of this part.

Proving part 3 essentially amounts to looking at two rules, [Com] and [Del$_{sub}$]. We start with the first one assuming

$$\frac{s_1 \xrightarrow{(p \bullet o) \rhd \bar{u}^{\ell_i}} s_1' \qquad s_2 \xrightarrow{(p \bullet o) \lhd \bar{n}^{\ell_o}} s_2' \qquad \mathcal{M}(\bar{u}, \bar{n}) = \sigma}{s_1 \mid s_2 \xrightarrow{p \bullet o \lfloor \sigma \rfloor \bar{u}^{\ell_i} \bar{n}^{\ell_o}} s_1' \mid s_2'}$$

and $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_1 \mid s_2$, hence $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_1$ and $\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_2$. We can thus apply the induction hypothesis on s_2 obtaining

$$\hat{\mathsf{R}}, \hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_2' \tag{4}$$

$$po\bar{n} \in \hat{\mathsf{K}} \tag{5}$$

From (5) and $\mathcal{M}(\bar{u},\bar{n}) = \sigma$ we deduce $po\bar{u}\cdot\sigma \in \hat{\mathsf{K}}$, which allows us to apply the induction hypothesis for s_1 yielding $\hat{\mathsf{R}},\hat{\mathsf{K}} \models^{s_\star}_{\sigma} s_1'$. Together with (4), this completes the proof of part 3 for rule [Com].

We are left to show part 3 for rule [Del_{sub}] that looks as follows:

$$\frac{s \xrightarrow{p\bullet o\lfloor\sigma\uplus[x\mapsto n]\rfloor\bar{u}^{\ell_i}\bar{n}^{\ell_o}} s'}{[x]^{\ell}s \xrightarrow{p\bullet o\lfloor\sigma\rfloor\bar{u}^{\ell_i}\bar{n}^{\ell_o}} s'\cdot[x\mapsto n]}$$

The proof of this part amount to showing the following:

$$\hat{\mathsf{R}},\hat{\mathsf{K}} \models^{s_\star}_{\sigma} [x]^{\ell}s \text{ implies } \hat{\mathsf{R}},\hat{\mathsf{K}} \models^{s_\star}_{\sigma\uplus[x\mapsto n]} s \tag{6}$$

$$\hat{\mathsf{R}},\hat{\mathsf{K}} \models^{s_\star}_{\sigma\uplus[x\mapsto n]} s' \text{ implies } \hat{\mathsf{R}},\hat{\mathsf{K}} \models^{s_\star}_{\sigma} s'\cdot[x\mapsto n] \tag{7}$$

Fact 6 allows us to apply the induction hypothesis, which yields exactly the assumption of (7). The application of (7) then concludes this case and the overall proof (since again the inductive cases are straightforward). We are left to show (6) since (7) is obvious from Definition 3. Observe that $x\#\mathsf{F}.\ell_i = \ell$. We can thus deduce (from the definition of Y_x in analysis rule [RRec] that $n \in \Pi_{x@\mathsf{L}.\ell'}(\hat{\mathsf{R}}.\ell')$ for all $\ell' \in \mathsf{F}.\ell$. This suffices to prove (6) and Lemma 2.