

# Relational Analysis for Delivery of Services <sup>★</sup>

Flemming Nielson, Hanne Riis Nielson, Jörg Bauer, Christoffer Rosenkilde  
Nielsen, and Henrik Pilegaard

Informatics and Mathematical Modelling, Technical University of Denmark  
Kongens Lyngby, Denmark.

{nielson,hrn,joba,crn,hepi}@imm.dtu.dk

**Abstract.** Many techniques exist for statically computing properties of the evolution of processes expressed in process algebras. Static analysis has shown how to obtain useful results that can both be checked and computed in polynomial time. In this paper we develop a static analysis in relational form which substantially improves the precision of the results obtained while being able to deal with the full generality of the syntax of processes. The analysis reveals a feasible complexity for practical examples and gives rise to a fast prototype. We use this prototype to automatically prove the correct delivery of messages for the implementation of an accident service, which is based on multiplexed communication, a crucial feature of global computing applications.

## 1 Introduction

Process algebras facilitate abstract models of a number of features of concurrent and distributed computation. Many use the notion of channel to provide end-to-end guarantees ensuring secure communications taking place. A prime example is the  $\pi$ -calculus [10] where channels can be freely created and guarantee that a message sent along the channel can only be received by a process listening on that channel. Indeed, processes not having access to the channel cannot observe or influence any properties of the values being sent along the channel. Hence end-to-end guarantees of proper delivery of messages is almost automatic.

Moving closer to the actual implementation level there is no direct counterpart of the notion of channels as used in the  $\pi$ -calculus although symmetric cryptosystems can be used to encode some of their properties. Practical techniques often include limiting the number of channels used and instead use multiplexing of several communications over a fixed set of channels. It is then a requirement on the transporting processes that they correctly implement the intended end-to-end communication. In this paper we use a running example, a version of the accident service taken from the automotive case study of the SENSORIA EU-project, where this problem arises.

This paper shows how to use static analysis for demonstrating that the service requests of the system are correctly distributed by the multiplexer process.

---

<sup>★</sup> This work has been partially sponsored by the project SENSORIA, IST-2005-016004.

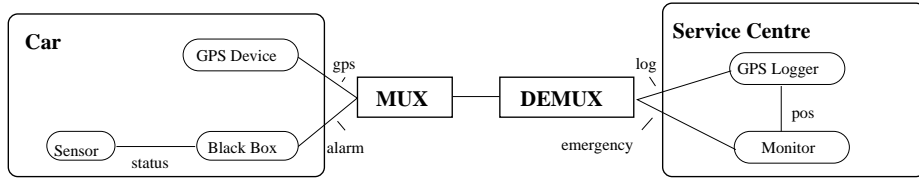
It turns out that so-called independent attribute analyses [11], as developed in e.g. [1], are unable to deliver the guarantees needed. A static analysis for multiplexing must be *relational*, that is, it must be able to capture precisely the dependence between various names. As an example, if a process contains output operations  $a\langle b, b \rangle$  and  $a\langle c, c \rangle$  then the analysis must be able to show the absence of the output operations  $a\langle b, c \rangle$  and  $a\langle c, b \rangle$ .

There are strong relational analyses around to prove multiplexing correct, most successful among them the abstract interpretations of Venet [16] and Feret [5]. In contrast to our proposed analyses, they are even able to distinguish different recursive instances of a process.

However, where our analysis is presented for standard  $\pi$ -calculus (with pattern matching) with the standard reduction semantics based on congruence, the analyses of [16, 5] rely on heavily customised versions of the language that include input-guarded replication, an instrumented semantics and explicit substitution environments. Thus, these customisations enable strong analyses, but are of disadvantage in the context of global computing where many calculi emerge, for instance [8] and [3], that all use the standard constructs of classical  $\pi$ , in particular congruence and reduction semantics. In contrast, our analysis, being based on these standards, can be easily transferred to new emerging languages, while [16, 5] would need to be completely re-designed. Beyond re-usability, which is of utmost importance for global computing, our proposal is implemented and enjoys a correctness result in terms of a subject reduction result, which is the standard proof technique. In contrast, [16, 5] settle for a very general soundness result of their abstract interpretation.

Our contribution is thus the development of a relational analysis that can be specified almost as naturally as the simpler (and in fact too simple) independent attribute analyses [1], that is easily extendable to new, emerging global computing calculi, because it relies on standard syntax, semantics, and proof technique. Indeed, our correctness results relies on the invariance of a correctness predicate under subject reduction, where the correctness predicate takes care of the implicit substitutions that need to be made very explicit—and hence deviating from standard reduction semantics—in the approaches of Venet [16] and Feret [5]. The analysis is implemented, and provides results that are sufficiently precise to validate our running example. While the worst case complexity is (necessarily) exponential we show that for realistic programs, e.g. for our running example, we are polynomial in solving the associated constraints. We should also point out that, due to our use of Alternation-free Least Fixed Point Logic (ALFP), the time needed for computing the best solution is asymptotically equal to the time needed for validating a solution (unlike approaches where validation is polynomial time but inference is nondeterministic polynomial time, the latter being exponential in practice).

*Outline.* We continue by presenting our running example, an accident service from the automotive case study of the SENSORIA EU-project. In Section 2, we present syntax and semantics of  $p\pi$ , our extension of the  $\pi$ -calculus with pattern



**Fig. 1.** The overall architecture of the accident service.

matching. The analysis itself is specified in Section 3, while its properties are reported in Section 4. Section 5 concludes.

### 1.1 The Accident Service.

A typical service-centred application is the *accident service*. The overall architecture of this service is depicted in Figure 1. In order to subscribe to this service, a car needs to be equipped with a GPS device and a black box. The black box frequently polls internal sensors for abnormal events. In such a case, it will start sending alarm messages containing the car’s and the driver’s identity. The service centre has two objectives: It logs the GPS data received from the car and monitors whether any alarm occurs. In that case, it gets the location of the car and the identity of its driver from the GPS logger and sends an SOS message to a rescue service (not modelled here). The somewhat intricate specification of the logger ensures that the most recently sent position is actually attached to the SOS.

All messages between the car and the service centre are communicated over a *multiplexed, wireless channel*— a feature typical of service oriented architectures. The multiplexer takes care of distributing messages correctly while providing optimal use of bandwidth.

We use a polyadic  $\pi$ -calculus,  $\mathfrak{p}\pi$ , extended with pattern matching in input prefixes (as in [2]) in order to write down the accident service formally (Table 1). For analysis purposes, action prefixes are annotated with *labels*. The usual precedence rules—parallelisation < summation < prefix, restriction, replication—hold for  $\mathfrak{p}\pi$  as well. An input prefix  $x(\bar{y}; \bar{u})$ , receives a tuple  $\bar{z}$  over channel  $x$  if the first  $|\bar{y}|$  elements of  $\bar{z}$  equal  $\bar{y}$ , thus binding the remaining elements to  $\bar{u}$ . The complete syntax and semantics are presented in Section 2. Our analysis presented in Section 3 will be able to find out that the MUX distributes messages correctly, that is, messages over the **emergency** channel will contain car and driver identity information only, whereas messages sent over the **log** channel will always contain car identities and position information.

## 2 The $\mathfrak{p}\pi$ -Calculus

### 2.1 Syntax

The syntax of the  $\pi$ -calculus extended with pattern matching,  $\mathfrak{p}\pi$ , can be seen in Table 2. As in the  $\pi$ -calculus we have *channels* that facilitate synchronous

GPS device:	$!(\nu \text{loc}) \text{gps}\langle \text{car}_i, \text{loc} \rangle^1$
Sensor:	$!(\text{status}\langle \text{car}_i, \text{ok} \rangle^2 + \text{status}\langle \text{car}_i, \text{crit} \rangle^3)$
Black Box:	$!\text{status}\langle \text{car}_i; x \rangle^4.[x = \text{crit}]^5.!\text{alarm}\langle \text{car}_i, \text{driver}_i \rangle^6$
GPS Logger:	$(\nu k)(!k(\cdot)^7.\text{log}(\cdot; y_{\text{car}}, y_{\text{pos}})^8.$ $(\text{pos}(y_{\text{car}}; \cdot)^9.\text{pos}\langle y_{\text{car}}, y_{\text{pos}} \rangle^{10}.k\langle \cdot \rangle^{11} + k\langle \cdot \rangle^{12})$ $  k\langle \cdot \rangle^{13})$
Monitor:	$!\text{emergency}(\cdot; z_{\text{car}}, z_d)^{14}.\text{pos}\langle z_{\text{car}} \rangle^{15}.\text{pos}\langle z_{\text{car}}; z_{\text{pos}} \rangle^{16}.\text{SOS}\langle z_{\text{car}}, z_{\text{pos}}, z_d \rangle^{17}$
Mux:	$!\text{gps}(\cdot; z_{\text{car}}, z_{\text{pos}})^{18}.\text{wifi}\langle \text{log}, z_{\text{car}}, z_{\text{pos}} \rangle^{19}$
Demux:	$!\text{alarm}(\cdot; z_{\text{car}}, z_d)^{20}.\text{wifi}\langle \text{emergency}, z_{\text{car}}, z_d \rangle^{21}$ $!\text{wifi}(\cdot; z_1, z_2, z_3)^{22}.z_1\langle z_2, z_3 \rangle^{23}$

**Table 1.** The  $\rho\pi$  specification of the accident service. The process  $P_{\text{acc}}$  is the parallel composition of the components stated here.

$P ::= 0$	Terminated Process
$  !P$	Replication
$  P_1   P_2$	Parallel
$  P_1 + P_2$	Choice
$  (\nu n)P$	Restriction
$  [x = y]^\ell P$	Match
$  x\langle \bar{y} \rangle^\ell.P$	Output
$  x(\bar{y}; \bar{u})^\ell.P$	Input

**Table 2.** Processes;  $P$

name passing communication. We use *names* picked from the denumerable set  $\text{Name}$  to denote channels and we shall use the notation  $n, m, p$  for elements of this set. Similarly, we shall assume a denumerable set  $\text{Var}$  of *variables* and let  $u, v$  range over this set. When necessary, we shall use  $x, y, z$  to range over the disjoint union  $\text{Name} \cup \text{Var}$ . However, as we shall see below names and variables are *bound* in different manners. The calculus is polyadic and we shall use bars to denote polyadic entities, e.g.  $\bar{n}, \bar{u}, \bar{x}$  etc.

The intuition behind the set of primitives is as follows: The inactive or *terminal process*,  $0$ , denotes the end of a process, a point from where no further progress can be made. The *parallel composition* construct,  $P_1 | P_2$ , represents the process that is a concurrent composition of two processes  $P_1$  and  $P_2$ . The *choice construct*,  $P_1 + P_2$ , is used to model non-deterministic behaviour. The *replication* construct,  $!P$ , describes a process that is the parallel composition of as many occurrences of  $P$  as necessary; in the scope of name passing this is adequate for expressing recursive behaviour. The *name restriction* construct,  $(\nu n)P$ , binds a

name,  $n$ , that may be used freely in  $P$ , but is not free in  $(\nu n)P$ , i.e. the scope of  $n$  is restricted to  $P$ . The guarded process,  $[x = y]P$ , has a simple 'if-then' behaviour - the execution of  $P$  can only commence if  $x$  and  $y$  denote the same name. The polyadic *output prefix* construct,  $x(\bar{y}).P$ , represents a process that desires to engage, as sender, in a synchronous exchange of information on the channel denoted by  $x$  and then proceed as described by  $P$ . However, the output can be completed only if a concurrent subprocess is simultaneously willing to participate, as receiver, in a *matching* communication on the same channel. The polyadic *input prefix* construct,  $x(\bar{y}; \bar{u}).P$ , represents a process that desires to engage, as receiver, in a synchronous exchange of information on the channel denoted by  $x$  and then proceed as  $P$ . The input can complete if:

1. a concurrent subprocess is simultaneously willing to engage, as sender, in a communication on the same channel, and
2. the output offered by this remote process *matches* the expectations expressed by the input pattern, i.e. the output and input vectors are both of length  $|\bar{y}| + |\bar{u}|$  and they agree on the names in the first  $|\bar{y}|$  positions.

If these conditions are satisfied the communication can commence binding each *variable* in  $\bar{u}$  to the *name* mentioned in the corresponding position of the output vector.

*Syntactic conventions.* As customary for the  $\pi$ -calculus we shall abstain from writing the terminal  $0$  at the end of example processes. Furthermore, we shall assume that well-formed programs do not contain free variables. However, for the convenience of writing examples we do allow free names. We use  $\star \in \{|\, +\}$  for brevity when parallel composition and choice are treated in the same way.

*Label Annotations.* To aid expressing the analysis in Section 3 we shall annotate the actions of the processes with *labels*  $\ell \in \mathbf{Lab}$  as in  $x(\bar{y})^\ell$ ,  $x(\bar{y}; \bar{u})^\ell$  and  $[x = y]^\ell$ . For simplicity we shall assume that the labels are *unique* in the process  $P_\star$  to be analysed. The labels play no role whatsoever in the semantics; they only serve as pointers into the syntax.

## 2.2 Semantics

We now give an operational semantics of the  $\mathbf{p}\pi$  calculus based on a *structural congruence*,  $\equiv$ , and a *reduction relation*,  $\rightarrow$ . This is a semantics in the style of Milner's reaction relation [9] for the original  $\pi$ -calculus. The resulting semantics clearly expresses an intuitive understanding of concurrency and interaction. The processes of  $\mathbf{p}\pi$  are grouped into congruence classes by the structural congruence relation, which is defined in Table 3. This definition ensures that the members of each class are congruent up to trivial syntactic restructuring. In the definition - and the following - we use  $\mathbf{fn}(P)$  and  $\mathbf{fv}(P)$  to denote the free names and free variables of the process  $P$ , respectively.

(NAM1) $(\nu n)(\nu m)P \equiv (\nu m)(\nu n)P$	(NAM2) $(\nu n)\mathbf{0} \equiv \mathbf{0}$
(NAM3) $(\nu n)(P \mid Q) \equiv (\nu n)P \mid Q$ if $n \notin \text{fn}(Q)$	
(ASSOC) $(P \star Q) \star R \equiv P \star (Q \star R)$	
(COM) $P \star Q \equiv Q \star P$	
(NIL) $P \mid \mathbf{0} \equiv P$	(REP) $!P \equiv P \mid !P$
(AEQ) $P \equiv_\alpha Q \Rightarrow P \equiv Q$	

**Table 3.** Structural congruence,  $P \equiv Q$ , is the smallest congruence relation on processes satisfying the axioms above. We use  $\star \in \{|\, +\}$  for brevity.

(ALPHA) $(\nu n)P \equiv_\alpha (\nu m)P[m/n]$ if $m \notin \text{fn}(P) \wedge [n] = [m]$
--

**Table 4.** Disciplined  $\alpha$ -equivalence;  $P \equiv_\alpha Q$ .

As usual the congruence includes  $\alpha$ -equivalence (Table 4) - asserting that processes are equivalent if they differ only in their choice of bound names. However, as we distinguish between names and variables and shall never substitute a variable for a name or variable we choose to define  $\alpha$ -equivalence only for names. Also, we write  $P[m/n]$  to denote the process that is as  $P$  except that every free occurrence of name  $n$  is replaced by name  $m$ ; the notion of substitution is formally defined in Table 5. Finally, we use the notion of *canonical names* ( $[n]$  as motivated and defined below) and demand that  $\alpha$ -equivalence only holds when the bound names have the same canonical name.

In the definition of substitution over a name restriction,  $\alpha$ -renaming is used to avoid name capture. This means that constants do not have representations that are stable under evaluation. However, syntactically unstable entities are not suitable for carrying static analysis information. Therefore we associate each constant  $n$  with a stable canonical name  $[n]$  and demand that  $\alpha$ -renaming be *disciplined*, such that canonical names are preserved, even when the syntactical representations change. Technically, the canonicalisation of names partitions the name-space into equivalence classes. Each canonical name uniquely identifies the defining syntactic occurrence giving rise to the associated class. Then  $\alpha$ -renaming (Rule (SRES) of Table 5) demands that new names be picked from appropriate classes.

The reductions of processes are given by the binary *reduction relation*, which is defined inductively as the least binary relation described by the axioms and rules of Table 6.

When the reduction relation holds between a pair of processes, written  $P \rightarrow P'$ , it means that  $P$  can evolve into  $P'$  by a single input/output reduction (COM) or a successful guard (MATCH) within some subprocess of  $P$ . The rule (COM) requires radices to be on a certain normal form and the rule (VAR) allows the use of the structural congruence for obtaining this form. The remaining rules, (PAR)

(SNIL)	$\mathbf{0}^{[m/z]}$	$= \mathbf{0}$
(SREP)	$(!P)^{[m/z]}$	$= !P^{[m/z]}$
(SPAR)	$(P_1 \star P_2)^{[m/z]}$	$= P_1^{[m/z]} \star P_2^{[m/z]}$
(SRES)	$((\nu n)P)^{[m/z]}$	$= \begin{cases} (\nu n)P & \text{if } z = n \\ (\nu n')(P^{[n'/n]})^{[m/z]} & \text{if } z \neq n \wedge m = n \wedge \\ & n' \notin \text{fn}(P) \wedge [n] = [n'] \\ (\nu n)P^{[m/z]} & \text{otherwise} \end{cases}$
(SMATCH)	$([x = y]P)^{[m/z]}$	$= [x^{[m/z]} = y^{[m/z]}]P^{[m/z]}$
(SOUT)	$(x\langle \bar{y} \rangle.P)^{[m/z]}$	$= x^{[m/z]}\langle \bar{y}^{[m/z]} \rangle.P^{[m/z]}$
(SIN)	$(x(\bar{y}; \bar{u}).P)^{[m/z]}$	$= \begin{cases} x^{[m/z]}(\bar{y}^{[m/z]}; \bar{u}).P & \text{if } z \in \{\bar{u}\} \\ x^{[m/z]}(\bar{y}^{[m/z]}; \bar{u}).P^{[m/z]} & \text{otherwise} \end{cases}$

**Table 5.** Substitution;  $P^{[m/z]}$ .

(MATCH)	$[n = n]P \rightarrow P$	(PAR)	$\frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}$	(CHO)	$\frac{P \rightarrow P'}{P + Q \rightarrow P'}$
(COM)	$m\langle n_1, \dots, n_k \rangle.P \mid m\langle n_1, \dots, n_j; u_{j+1}, \dots, u_k \rangle.Q \rightarrow$ $P \mid Q^{[n_{j+1}/u_{j+1}]} \dots [n_k/u_k]$				
(VAR)	$\frac{P \equiv Q \quad Q \rightarrow Q' \quad Q' \equiv P'}{P \rightarrow P'}$		(RES)	$\frac{P \rightarrow P'}{(\nu n)P \rightarrow (\nu n)P'}$	

**Table 6.** Reduction Semantics;  $P \rightarrow P'$ .

and (RES), simply propagate reductions across parallel composition and name restrictions respectively, while (CHO) lets one process in a summation proceed.

### 3 Relational Analysis

We set the stage for the analysis by defining some auxiliary information. First, a *label environment*  $\mathbf{L}$  is defined as a mapping

$$\mathbf{L} : \mathbf{Lab} \hookrightarrow \mathbf{Var}^*$$

that to each label  $\ell$  associates a sequence  $\bar{u}$  of variables that have been introduced *before* this point in the process. More formally, we shall take  $\mathbf{L} = \mathcal{L}_\epsilon \llbracket P_\star \rrbracket$  where  $\mathcal{L}_{\bar{u}}$  (for  $\bar{u} \in \mathbf{Var}^*$ ) is defined in Table 7. Here we write  $\uplus$  for joining two mappings with disjoint domains and  $[\ ]$  for the mapping with empty domain. The notation  $\bar{u}\bar{v}$  stands for the concatenation of  $\bar{u}$  and  $\bar{v}$ . We will use the notation  $\langle x_1, \dots, x_r \rangle$  to write down vectors.

In the analysis we shall also need a representation of the *flow of control* in the process  $P_\star$  of interest. We shall represent this by a *flow mapping*  $\mathbf{F}$  that to each label  $\ell$  associates the set of labels that will become *exposed* once the action labelled  $\ell$  has been executed; thus

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

$\mathcal{L}_{\bar{u}}[P_1 \star P_2] = \mathcal{L}_{\bar{u}}[P_1] \uplus \mathcal{L}_{\bar{u}}[P_2]$	$\mathcal{L}_{\bar{u}}[!P] = \mathcal{L}_{\bar{u}}[P]$
$\mathcal{L}_{\bar{u}}[(\nu n)P] = \mathcal{L}_{\bar{u}}[P]$	$\mathcal{L}_{\bar{u}}[\mathbf{0}] = []$
$\mathcal{L}_{\bar{u}}[x(\bar{y})^\ell.P] = \mathcal{L}_{\bar{u}}[P][\ell \mapsto \bar{u}]$	$\mathcal{L}_{\bar{u}}[x(\bar{y}; \bar{v})^\ell.P] = \mathcal{L}_{\bar{u}\bar{v}}[P][\ell \mapsto \bar{u}]$
$\mathcal{L}_{\bar{u}}[[x = y]^\ell P] = \mathcal{L}_{\bar{u}}[P][\ell \mapsto \bar{u}]$	

**Table 7.** Label environment.

$\mathcal{F}[P_1 \star P_2] = \text{let } (F_1, E_1) = \mathcal{F}[P_1]$
$(F_2, E_2) = \mathcal{F}[P_2]$
$\text{in } (F_1 \uplus F_2, E_1 \cup E_2)$
$\mathcal{F}[!P] = \mathcal{F}[P]$
$\mathcal{F}[(\nu n)P] = \mathcal{F}[P]$
$\mathcal{F}[\mathbf{0}] = ([], \emptyset)$
$\mathcal{F}[x\langle y_1, \dots, y_k \rangle^\ell.P] = \text{let } (F, E) = \mathcal{F}[P]$
$\text{in } (F[\ell \mapsto E], \{\ell\})$
$\mathcal{F}[x\langle y_1, \dots, y_j; u_{j+1}, \dots, u_k \rangle^\ell.P] = \text{let } (F, E) = \mathcal{F}[P]$
$\text{in } (F[\ell \mapsto E], \{\ell\})$
$\mathcal{F}[[x = y]^\ell P] = \text{let } (F, E) = \mathcal{F}[P]$
$\text{in } (F[\ell \mapsto E], \{\ell\})$

**Table 8.** Flow information;  $\mathcal{F}[P]$ .

The function  $\mathcal{F}$  defined in Table 8 will for each process  $P$  define such a mapping together with the set of *exposed labels* of the process itself and we shall define  $(F, E) = \mathcal{F}[P_\star]$ .

*Example 1.* The annotated running example,  $P_{acc}$ , was given in Table 1. Using the functions  $\mathcal{L}$  and  $\mathcal{F}$  we obtain  $E = \{1, 2, 3, 4, 7, 13, 14, 18, 20, 22\}$  and the following samples for flow  $F$  and label environment  $L$ .

$\ell$	1	4	8	14	15	16	17
$F.\ell$	$\emptyset$	$\{5\}$	$\{9, 12\}$	$\{15\}$	$\{16\}$	$\{17\}$	$\emptyset$
$L.\ell$	$\epsilon$	$\epsilon$	$\epsilon$	$\epsilon$	$\langle z_{car}, z_d \rangle$	$\langle z_{car}, z_d \rangle$	$\langle z_{car}, z_d, z_{pos} \rangle$

### 3.1 Analysis domains

The *abstract environments*  $\hat{R}$  of the analysis will, given a label  $\ell$ , return a set of sequences of names; the structure of these sequences will equal that of  $L.\ell$  and will determine the potential values of the variables. Thus we shall take:

$$\hat{R} : \mathbf{Lab} \rightarrow \mathcal{P}(\mathbf{Name}^*)$$



Note that this is exactly the point, where the analysis becomes relational. We record sets of tuples of names, which are received at the same time, rather than tuples of sets as it would be the case for an independent attribute analysis.

If  $\hat{R}.\ell = \emptyset$  it means that the program point  $\ell$  is not reachable; if  $\hat{R}.\ell = \{\epsilon\}$  it means that no variables are bound at that program point.

We use the following auxiliary function to determine the potential values of a variable  $u$  at the label  $\ell$ :  $\Pi_{u@L.\ell}(\bar{w})$ , where  $\bar{w}$  has the same length as  $L.\ell$  and  $u \in \{L.\ell\}$ . We shall use  $\bar{w}$  to denote elements of  $\hat{R}.\ell$ , that is, vectors of names. In the cases where more than one occurrence of  $u$  occurs in  $L.\ell$  we always select the rightmost—corresponding to the most recently bound one.

Given an element  $\bar{w} \in \hat{R}.\ell$  we can now determine the *value*  $\Pi_{x@L.\ell}(\bar{w})$  of  $x$ . We have two cases depending on whether  $x$  is a name or a variable:

$$\begin{aligned} \Pi_{n@L.\ell}(\bar{w}) &= n \quad (\text{for } n \text{ being a name}) \\ \Pi_{u@L.\ell}(\bar{w}) &= n_s \text{ where } \bar{w} = \langle n_1, \dots, n_r \rangle, L.\ell = \langle u_1, \dots, u_r \rangle \\ &\quad \text{and } s = \max\{i \mid u_i = u\} \end{aligned}$$

This operation is extended to sets  $R$  of sequences of names and to sequences  $\bar{x}$  of names and variables as in  $\Pi_{\bar{x}@L.\ell}(R)$ .

### 3.2 The relational analysis

The *judgements* of the analysis have the form  $\hat{R}, \hat{K} \vdash_{L,F} P$  where  $L, F$  and  $\hat{R}$  are as above and

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

records the tuples that potentially are communicated over the channels. Table 9 defines the judgements. Intuitively, they define, whether a given pair  $\hat{R}, \hat{K}$  is a valid analysis result. Hence, Table 9 specifies a set of solutions. As discussed in Section 4.2, we are interested in and able to compute the most precise one.

The rules (RPAR), (RREP) and (RRES) are straightforward as they only require that the subprocesses can be analysed using the same analysis information. The rule (RNIL) expresses that any analysis information will be correct for 0.

In the rule (ROUT) we write  $X$  for the set of sequences of names that take part in the communication; this set is obtained by extracting the names corresponding to  $xy_1 \dots y_k$  from  $\hat{R}.\ell$  using  $L.\ell$  as expressed by  $\Pi_{x\bar{y}@L.\ell}(\hat{R}.\ell)$ . The condition  $X \subseteq \hat{K}$  ensures that the output is recorded in  $\hat{K}$  and the premise of the rule expresses a reachability test in that the analysis information is only required to be valid for the continuation if something might be communicated over the channel. Finally, the side condition  $\forall \ell' \in F.\ell : \hat{R}.\ell \subseteq \hat{R}.\ell'$  requires that the information of  $\hat{R}.\ell$  flows to all the program points that follow directly after  $\ell$ .

In the rule (RIN) we use the set  $X$  to capture how the environment  $\hat{R}.\ell$  is extended to contain the bindings of the new variables  $\bar{u}$ . The set is constructed by first selecting those sequences  $\bar{m}\bar{p}$  from  $\hat{K}$  that match the potential values of  $x$  and  $\bar{y}$  in some  $\bar{w}$  of  $\hat{R}.\ell$  and then extending those  $\bar{w}$  sequences with  $\bar{p}$ . The sequences  $X$  constructed in this way will then be the possible environments at

$\text{(RPAR)} \quad \frac{\hat{R}, \hat{K} \vdash_{L,F} P_1 \quad \hat{R}, \hat{K} \vdash_{L,F} P_2}{\hat{R}, \hat{K} \vdash_{L,F} P_1 \star P_2}$	$\text{(RREP)} \quad \frac{\hat{R}, \hat{K} \vdash_{L,F} P}{\hat{R}, \hat{K} \vdash_{L,F} !P}$
$\text{(RRES)} \quad \frac{\hat{R}, \hat{K} \vdash_{L,F} P}{\hat{R}, \hat{K} \vdash_{L,F} (\nu n)P}$	$\text{(RNIL)} \quad \hat{R}, \hat{K} \vdash_{L,F} 0$
$\text{(ROUT)} \quad \frac{X \neq \emptyset \Rightarrow \hat{R}, \hat{K} \vdash_{L,F} P}{\hat{R}, \hat{K} \vdash_{L,F} x\langle y \rangle^\ell . P} \quad \text{if} \quad \begin{array}{l} X \subseteq \hat{K} \wedge \forall \ell' \in F.\ell : X \neq \emptyset \Rightarrow \hat{R}.\ell \subseteq \hat{R}.\ell' \\ \text{where } X = \Pi_{x\bar{y}@\mathbb{L}.\ell}(\hat{R}.\ell) \end{array}$	
$\text{(RIN)} \quad \frac{X \neq \emptyset \Rightarrow \hat{R}, \hat{K} \vdash_{L,F} P}{\hat{R}, \hat{K} \vdash_{L,F} x(\bar{y}; \bar{u})^\ell . P}$ <p style="margin-left: 20px;">         if <math>\forall \ell' \in F.\ell : X \subseteq \hat{R}.\ell'</math>          where <math>X = \{\bar{w}\bar{p} \mid \bar{w} \in \hat{R}.\ell \wedge \bar{m} = \Pi_{x\bar{y}@\mathbb{L}.\ell}(\bar{w}) \wedge \bar{m}.\bar{p} \in \hat{K} \wedge  \bar{p}  =  \bar{u} \}</math> </p>	
$\text{(RMATCH)} \quad \frac{X \neq \emptyset \Rightarrow \hat{R}, \hat{K} \vdash_{L,F} P}{\hat{R}, \hat{K} \vdash_{L,F} [x = y]^\ell . P} \quad \text{if} \quad \forall \ell' \in F.\ell : X \subseteq \hat{R}.\ell'$ <p style="margin-left: 20px;">         where <math>X = \{\bar{w} \in \hat{R}.\ell \mid \Pi_{x@\mathbb{L}.\ell}(\bar{w}) = \Pi_{y@\mathbb{L}.\ell}(\bar{w})\}</math> </p>	

**Table 9.** Relational Analysis;  $\hat{R}, \hat{K} \vdash_{L,F} P$ .

all the program points that follow directly after  $\ell$ . Note that the continuation  $P$  is only analysed if  $X$  is non-empty, that is if there actually are some sequences in  $\hat{K}$  that satisfy the conditions expressed in the definition of  $X$ .

Finally, in the rule (RMATCH) the set  $X$  is defined to be those sequences from  $\hat{R}.\ell$  that agree on the values of  $x$  and  $y$  as obtained using the positions obtained from  $L.\ell$ . Only these sequences are required to be recorded as possible environments in the program points that follow directly after  $\ell$  and the continuation  $P$  will only be analysed if  $X$  is non-empty, that is, when the test might indeed be passed.

*Example 2.* For the running example  $P_{acc}$  the following choice of  $\hat{R}$  and  $\hat{K}$  satisfies  $\hat{R}, \hat{K} \vdash_{L,F} P_{acc}$  (and is indeed the most precise solution). Note that this results holds for an arbitrary but fixed number  $n > 0$  of identically defined cars.

$$\hat{K} = \{ \langle \text{pos}, \text{car}_i, \text{loc} \rangle, \langle \text{emergency}, \text{car}_i, \text{driver} \rangle, \langle \text{log}, \text{car}_i, \text{loc} \rangle, \langle \text{alarm}, \text{car}_i, \text{driver} \rangle, \\ \langle \text{status}, \text{car}_i, \text{crit} \rangle, \langle \text{status}, \text{car}_i, \text{ok} \rangle, \langle \text{gps}, \text{car}_i, \text{loc} \rangle, \langle k \rangle, \langle \text{pos}, \text{car} \rangle, \\ \langle \text{SOS}, \text{car}_i, \text{loc}, \text{driver} \rangle, \langle \text{wifi}, \text{emergency}, \text{car}_i, \text{driver} \rangle, \langle \text{wifi}, \text{log}, \text{car}_i, \text{loc} \rangle \\ | i = 1, \dots, n \}$$

$$\begin{array}{llll} \hat{R}.5: \{ \langle \text{ok} \rangle, \langle \text{crit} \rangle \} & \hat{R}.6: \{ \langle \text{crit} \rangle \} & \hat{R}.9: \{ \langle \text{car}_i, \text{loc} \rangle \} & \hat{R}.10: \{ \langle \text{car}_i, \text{loc} \rangle \} \\ \hat{R}.11: \{ \langle \text{car}_i, \text{loc} \rangle \} & \hat{R}.12: \{ \langle \text{car}_i, \text{loc} \rangle \} & \hat{R}.15: \{ \langle \text{car}_i, \text{driver} \rangle \} & \\ \hat{R}.16: \{ \langle \text{car}_i, \text{driver} \rangle \} & \hat{R}.17: \{ \langle \text{car}_i, \text{driver}, \text{loc} \rangle \} & \hat{R}.19: \{ \langle \text{car}_i, \text{loc} \rangle \} & \\ \hat{R}.21: \{ \langle \text{car}_i, \text{driver} \rangle \} & \hat{R}.23: \{ \langle \text{log}, \text{car}_i, \text{loc} \rangle, \langle \text{emergency}, \text{car}_i, \text{driver} \rangle \} & & \end{array}$$

For all labels  $\ell \in \{1, 2, 3, 4, 7, 8, 13, 14, 18, 20, 22\}$  we take  $\hat{R}.\ell = \{\epsilon\}$ . For brevity, we left out the condition  $i = 1, \dots, n$ , when stating the  $\hat{R}.\ell$  sets.

As stated in the introduction, we see from  $\hat{K}$  that only car and driver identity are communicated over the emergency channel and that messages sent over log only contain car and position information. Moreover, information from different cars is not mixed up. This can be inferred regardless of the number of cars involved, which would be impossible given a non-relational analysis. If we changed the specification of the black box to *separately* send car and driver information, then our analysis would detect the (real) error, that car and driver information from different cars may be mixed up in the multiplexer.

## 4 Properties of the Analysis

### 4.1 Correctness

The correctness of the analysis is formulated in terms of a *correctness predicate* which is shown invariant under subject reduction in Theorem 1. In contrast, mere analysability is not preserved under subject reduction (c.f. Appendix A). The correctness predicate is defined in Definition 1 and relates a process syntactically to the process initially analysed thus taking care of the implicit substitutions prevalent in standard reduction semantics.

In the following we will use small Greek letters to denote input, output, and match prefixes. Moreover, we shall assume that  $P_\star$  is a uniquely labelled process, and we shall fix  $L = \mathcal{L}_\epsilon[[P_\star]]$  and  $(F, E) = \mathcal{F}[[P_\star]]$  as information derived from  $P_\star$ . Additionally, assume a subexpression  $\alpha^\ell.P'$  of  $P_\star$ , an analysis result  $\hat{R}, \hat{K} \vdash_{L,F} P_\star$  and an element  $\bar{w} \in \hat{R}.\ell$ . We define the instantiation of  $\alpha^\ell.P'$  with  $\bar{w}$ , written  $\alpha^\ell.P'[\bar{w}]$  to be the process<sup>1</sup>

$$\alpha^\ell.P'[\Pi_{u_r \oplus L.\ell}(\bar{w})/u_r] \cdots [\Pi_{u_1 \oplus L.\ell}(\bar{w})/u_1]$$

where  $u_1, \dots, u_r$  are the variables in  $L.\ell$ . We define the correctness predicate for a process  $Q$  as follows.

**Definition 1 (Correctness Predicate).** *Correctness predicate*  $\hat{R}, \hat{K} \models^{P_\star} Q$  holds if and only if:

1.  $\hat{R}, \hat{K} \vdash_{L,F} P_\star$
2.  $\forall \ell \in E : \epsilon \in \hat{R}.\ell$
3. For all  $\beta^\ell.Q'$  exposed in  $Q$ , there exists a subexpression  $\alpha^\ell.P'$  of  $P_\star$  and a  $\bar{w} \in \hat{R}.\ell$ , such that  $\hat{R}, \hat{K} \vdash_{L,F} \alpha^\ell.P'$  and  $\alpha^\ell.P'[\bar{w}] \equiv \beta^\ell.Q'$ .

<sup>1</sup> The instantiation applies the cumulative effect of all implicit substitutions  $[\Pi_{u_r \oplus L.\ell}(\bar{w})/u_r] \cdots [\Pi_{u_1 \oplus L.\ell}(\bar{w})/u_1]$  that have taken place during reduction—hence unlike the approaches of Venet [16] and Feret [5] we do not need to modify the standard reduction semantics to use explicit substitutions (nor to rely on a customised version of the semantics).

First, we observe some auxiliary properties, whose proofs are omitted. Note that conditions (1) and (2) of Definition 1 imply (3) when reasoning about  $P_\star$ .

**Lemma 1 (Initial Process).** *For all processes  $P_\star$  with L, F, and E as above we have: If  $\hat{R}, \hat{K} \vdash_{L,F} P_\star$  and  $\epsilon \in \hat{R}.l$  for all  $l \in E$ , then  $\hat{R}, \hat{K} \models^{P_\star} P_\star$ .*

Together with Theorem 1, which states the invariance of the correctness predicate under the reduction relation, this lemma can be used to show that any process derived from the initial process  $P_\star$  by the transitive closure of the reduction relation is structurally congruent to a subprocess  $P'$  of  $P_\star$ , where each variable of  $P'$  is substituted by one name predicted by the analysis. This constitutes the *correctness of our relational analysis*.

Before getting to the correctness theorem, we state some lemmas about exposed subexpressions with respect to structural congruence and valid analysis results.

**Lemma 2.** *Let  $P \equiv Q$  be two processes. For all  $\alpha^\ell.P'$  exposed in  $P$ , there exists a  $\beta^\ell.Q'$  exposed in  $Q$  such that  $\alpha^\ell.P' \equiv \beta^\ell.Q'$ .*

**Lemma 3.** *If  $\alpha^\ell.P'$  is exposed in  $P$  and  $\hat{R}, \hat{K} \vdash_{L,F} P$ , then  $\hat{R}, \hat{K} \vdash_{L,F} \alpha^\ell.P'$ .*

The validity of the correctness predicate is invariant under structural congruence as formalised by the following lemma.

**Lemma 4.** *Let  $P_\star, Q$ , and  $R$  be processes. If  $Q \equiv R$  and  $\hat{R}, \hat{K} \models^{P_\star} Q$ , then  $\hat{R}, \hat{K} \models^{P_\star} R$ .*

*Proof.* Let  $\gamma^\ell.R'$  be exposed in  $R$ . As  $Q \equiv R$ , there exists  $\beta^\ell.Q'$  exposed in  $Q$ . As  $\hat{R}, \hat{K} \models^{P_\star} Q$ , we know that there exists a  $\alpha^\ell.P'$  subexpression of  $P_\star$  and a  $\bar{w} \in \hat{R}.l$  such that  $\hat{R}, \hat{K} \vdash_{L,F} \alpha^\ell.P'$  and  $\alpha^\ell.P'[\bar{w}] \equiv \beta^\ell.Q'$ . By transitivity of  $\equiv$ , we obtain  $\alpha^\ell.P'[\bar{w}] \equiv \gamma^\ell.R'$  and thus  $\hat{R}, \hat{K} \models^{P_\star} R$ . This concludes the proof of Lemma 4.

The validity of the correctness predicate is preserved under reduction as formalised by the following theorem.

**Theorem 1 (Subject Reduction).** *Let  $P_\star$  be a process. If  $Q \rightarrow R$  and  $\hat{R}, \hat{K} \models^{P_\star} Q$  then  $\hat{R}, \hat{K} \models^{P_\star} R$ .*

*Proof.* The proof is by induction on the inference of  $Q \rightarrow R$ . First consider the rule (COM)

$$m(\bar{n}\bar{p})^{\ell_0}.Q_0 \mid m(\bar{n}; \bar{u})^{\ell_1}.Q_1 \rightarrow Q_0 \mid Q_1[\bar{p}/\bar{u}]$$

and assume that there exists  $\alpha_0^{\ell_0}.P_0, \alpha_1^{\ell_1}.P_1$  subprocesses of  $P_\star$  as well as  $\bar{w}_i \in \hat{R}.l_i$  for  $i = 0, 1$  such that

$$\hat{R}, \hat{K} \vdash_{L,F} \alpha_0^{\ell_0}.P_0 \tag{1}$$

$$\hat{R}, \hat{K} \vdash_{L,F} \alpha_1^{\ell_1}.P_1 \tag{2}$$

$$(\alpha_0^{\ell_0}.P_0)[\bar{w}_0] \equiv m(\bar{n}\bar{p})^{\ell_0}.Q_0 \tag{3}$$

$$(\alpha_1^{\ell_1}.P_1)[\bar{w}_1] \equiv m(\bar{n}; \bar{u})^{\ell_1}.Q_1 \tag{4}$$

We need to prove that for all  $\beta^\ell.Q'$  exposed in  $Q_0 \mid Q_1[\bar{p}/\bar{u}]$ , there exists a  $\alpha^\ell.P'$  subexpression of  $P_\star$  and a  $\bar{w} \in \hat{R}.\ell$ , such that (a)  $\hat{R}, \hat{K} \vdash_{L,F} \alpha^\ell.P'$  and (b)  $\alpha^\ell.P'[\bar{w}] \equiv \beta^\ell.Q'$ .

**Case 1.** Let  $\beta^\ell.Q'$  be exposed in  $Q_0$  and let  $\alpha_0^{\ell_0} = x_0 \langle \bar{y}_0 \bar{z}_0 \rangle$ . By the definition of F, we get

$$\ell \in F.\ell_0 \quad (5)$$

From (3) we obtain  $\Pi_{x_0 \bar{y}_0 \bar{z}_0 @ L.\ell_0}(\bar{w}_0) = m\bar{n}\bar{p}$ . Together with (1), the definition of (ROUT), and (5) this yields:

$$m\bar{n}\bar{p} \in \hat{K} \quad (6)$$

$$\hat{R}, \hat{K} \vdash_{L,F} P_0 \quad (7)$$

$$\bar{w}_0 \in \hat{R}.\ell \quad (8)$$

Choose  $\bar{w} = \bar{w}_0$ . Requirement (a) is then proven using (7) with Lemma 3. Requirement (b) follows from (3), (8) using Lemma 2.

**Case 2.** Let  $\beta^\ell.Q'$  be exposed in  $Q_1[\bar{p}/\bar{u}]$  and let  $\alpha_1^{\ell_1} = x_1(\bar{y}_1; \bar{u})^{\ell_1}$ . Let  $\beta^\ell.Q'$  be exposed in  $Q_1[\bar{p}/\bar{u}]$  implying

$$\ell \in F.\ell_1 \quad (9)$$

From (4) we obtain  $\Pi_{x_1 \bar{y}_1 @ L.\ell_1}(\bar{w}_1) = m\bar{n}$ . Using this fact together with (9) and (6) we can apply rule (RIN) and obtain:

$$\bar{w}_1 \bar{p} \in \hat{R}.\ell \quad (10)$$

$$\hat{R}, \hat{K} \vdash_{L,F} P_1 \quad (11)$$

We choose  $\bar{w} = \bar{w}_1 \bar{p}$  which adheres to requirement (b) by (10). Requirement (a) is clear using Lemma 3 and (11). For requirement (b), we deduce  $P_1[\bar{w}_1] \equiv Q_1$  from (4). By the definition of F, we obtain  $L.\ell = (L.\ell_1)\bar{u}$ . Together with the definition of instantiating an expression with an analysis result we get  $P_1[\bar{w}] \equiv Q_1[\bar{p}/\bar{u}]$ . An application of Lemma 2 to this concludes the proof of the case for (COM).

We shall now consider the application of the rule (MATCH), that is,

$$[n = n]^{\ell_0}.Q_0 \rightarrow Q_0$$

We know that there exists  $[x_0 = y_0]^{\ell_0}.P_0$  subexpression of  $P_\star$  and  $\bar{w}_0 \in \hat{R}.\ell_0$  such that

$$\hat{R}, \hat{K} \vdash_{L,F} [x_0 = y_0]^{\ell_0}.P_0 \quad (12)$$

$$([x_0 = y_0]^{\ell_0}.P_0)[\bar{w}_0] \equiv [n = n]^{\ell_0}.Q_0 \quad (13)$$

Let  $\beta^\ell.Q'$  be exposed in  $Q_0$ . We need to show that there exists a  $\bar{w} \in \hat{R}.\ell$  and a subexpression  $\alpha^\ell.P'$  of  $P_\star$  such that (c)  $\hat{R}, \hat{K} \vdash_{L,F} \alpha^\ell.P'$  and (d)  $(\alpha^\ell.P')[\bar{w}] \equiv \beta^\ell.Q'$ . As  $\beta^\ell.Q'$  is exposed in  $Q_0$ , we deduce

$$\ell \in F.\ell_0 \quad (14)$$

By (13), we may deduce

$$\Pi_{x_0 @ \ell_0}(\bar{w}_0) = \Pi_{y_0 @ \ell_0}(\bar{w}_0) \quad (15)$$

$$P_0[\bar{w}_0] \equiv Q_0 \quad (16)$$

We can apply rule (RMATCH) with (15), (14), and (12) to obtain  $\bar{w}_0 \in \hat{R}.\ell$  and  $\hat{R}, \hat{K} \vdash_{L,F} P_0$ . If we choose  $\bar{w} = \bar{w}_0$ , we obtain (c) and (d) from  $\hat{R}, \hat{K} \vdash_{L,F} P_0$  and (16) using Lemma 3 and Lemma 2.

The result for an application of (VAR) is an immediate consequence of the induction hypothesis and Lemma 4. Let us now consider rule (PAR), that is,

$$\frac{Q \rightarrow Q'}{Q \mid R \rightarrow Q' \mid R}$$

and assume  $\hat{R}, \hat{K} \models^{P^*} Q \mid R$ . Obviously, this implies  $\hat{R}, \hat{K} \models^{P^*} Q$  and  $\hat{R}, \hat{K} \models^{P^*} R$  separately. By the induction hypothesis, we obtain  $\hat{R}, \hat{K} \models^{P^*} Q'$  and hence  $\hat{R}, \hat{K} \models^{P^*} Q' \mid R$ . This argumentation can be applied analogously to rule (RES), because the exposed expressions of  $(\nu n)Q$  are just those of  $Q$ . Also, the case (CHO) is completely analogous. This concludes the proof of Theorem 1.

## 4.2 Implementation

We have implemented our relational analysis in the Succinct Solver [13], which is able to efficiently compute the stable model of an expressive fragment of predicate logic. From the analysis specification for a program  $P_*$ , we generate a clause  $\varphi_{P_*}$  such that  $\hat{R}, \hat{K} \vdash_{L,F} P_*$  if and only if  $\vdash \varphi_{P_*}$ . Each model of the clause corresponds to an analysis solution. The generated clauses belong to the class of Alternation-free Least Fixpoint Logic (ALFP) described in [15]. Proposition 1 of [15] states that the set of all solutions of an ALFP clause always has a least element corresponding to the least analysis result we aim for. The Succinct Solver computes this least solution.

*Example 3.* Consider the specification of the Black Box in Table 1:

$$! \text{status}(\text{car}_i; x)^4. [x = \text{crit}]^5. ! \text{alarm}(\text{car}_i)^6$$

The clause generated for this excerpt according to rules (RIN), (RMATCH) and (ROUT) will essentially look as follows:

$$\begin{aligned} \exists u. \hat{K}(\text{status}, \text{car}_i, u) \Rightarrow ( & \\ (\forall u. \hat{K}(\text{status}, \text{car}_i, u) \Rightarrow \hat{R}.5(u)) \wedge & \\ \exists u'. (\hat{R}.5(u') \wedge u' = \text{crit}) \Rightarrow ( & \\ (\forall u'. \hat{R}.5(u) \wedge u' = \text{crit} \Rightarrow \hat{R}.6(u')) \wedge & \\ \hat{K}(\text{alarm}, \text{car}_i)) & \end{aligned}$$

Each (canonical) name corresponds to a constant in the clause. For each label  $\ell$  in the program there is a  $|L.\ell|$ -ary relation  $\hat{R}.\ell$ . Furthermore, for each message

length  $m$ , there is an  $m + 1$ -ary relation  $\hat{K}$ . The existential parts of the clause take care of the various reachability conditions of the form  $X \neq \emptyset$ , whereas the universal parts take care of letting the information gathered in  $X$  flow to the right places. As an aid to readability we have assumed that  $\hat{R}.4 = \{\epsilon\}$  thus not generated any formula involving  $\hat{R}.4$ .

### 4.3 Complexity

There are mainly three quantities determining the complexity of solving the ALFP clause that is generated for a program  $P_\star$ . First, the size  $n$  of  $P_\star$ ; second, the maximal nesting depth of variables bounded by  $m = \max_{\ell \in \mathbf{Lab}} |\mathbf{L}.\ell|$ ; third, the length of messages bounded by  $k = \max_{x(\bar{y}) \in P_\star} |\bar{y}|$ . Quantities  $m$  and  $k$  determine the maximal arity of relations, whereas  $n$  bounds the size of the universe over which the clause is evaluated. The maximal nesting depth of quantifiers is bounded by  $m$ . Altogether, we can apply Proposition 1 of [14], which itself makes use of the algorithm presented in [4], to obtain a complexity bound of  $\mathcal{O}(n^{3+k+m})$  for solving the clause generated from  $P_\star$ .

This complexity may be exponential in the worst-case. However, the worst-case is only realised for programs, where the number of sequenced input prefixes (each one binding new variables) and/or the arity of sent tuples grows linearly with the program. In contrast, we may observe that for typical programs, where processes often consist of reception-processing-reply,  $m$  and  $k$  may be considered constants rendering the complexity polynomial in the size of the program (with a rather large exponent, though). For the running example and other examples of similar size, the least solution to the analysis problem could be computed in less than a second.

## 5 Conclusion

The proper modelling of services for global computing necessitates the ability to model services in process algebras without using primitives (like the dynamic creation of very flexible channels as in the  $\pi$ -calculus) that have no direct counterpart in actual systems. We used a running example based on a multiplexing device part of the accident service used in the automotive case study of the SENSORIA EU-project. This increases the difficulty of validating that the models enjoy the desired properties and at the same time calls for the use of automatic analysis techniques to deal with the scalability issues of “realistic” models.

In this paper we have developed a new relational analysis for a  $\pi$ -calculus extended with pattern matching. The core benefit of a relational analysis (in contrast to an independent attribute analysis) is that sets of tuples of names being received at the same time are tracked (in contrast to tuples of sets). If a process contains output operations  $a\langle \mathbf{b}, \mathbf{b} \rangle$  and  $a\langle \mathbf{c}, \mathbf{c} \rangle$  then a relational analysis is able to show the absence of the output operations  $a\langle \mathbf{b}, \mathbf{c} \rangle$  and  $a\langle \mathbf{c}, \mathbf{b} \rangle$ —while an independent attribute analysis is not.

We have shown that semantic correctness amounts to the invariance of a correctness predicate under subject reduction. Furthermore, we have shown that the analysis has polynomial time complexity on realistic programs. We have used the analysis to validate the correct delivery of services in our running example.

In future work we plan to transfer the analysis technology to the richer set of process calculi being developed in the SENSORIA EU-project for describing the behaviour of services. Examples are likely to include variations of the constructs presented in [8, 6, 3] and [7]. For that work, we will benefit from the fact that we rely on full standard syntax and reduction semantics of  $\pi$ , on which [8, 6, 3, 7] are all based. We also plan to investigate the feasibility of using annotations to indicate which binding occurrences demand a relational treatment. This could lead to developing a mixed independent-attribute and relational analysis. The advantage of such an analysis would be that essentially cubic-time [12] methods for independent-attribute analyses could be used except for those cases where a truly relational analysis (of higher time complexity) is needed. Finally, we plan to incorporate techniques that can tell distinct recursive instances, e.g. several cars having an accident at the same time, apart, i.e., allow the analysis to find out that the SOS messages contain the correct position of each car (and not the position of another car).

**Acknowledgements.** The design of the  $p\pi$  calculus was part of a group effort by the following members of the Language Based Technology research group at their retreat in Schloss Dagstuhl: C. R. Nielsen, F. Nielson, H. Pilegaard, C. Probst, H. Riis Nielson, T. Tolstrup, and Y. Zhang.

## References

1. C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static analysis for the  $\pi$ -calculus with applications to security. *Information and Computation*, 168:68–92, 2001.
2. Chiara Bodei, Mikael Buchholtz, Pierpaolo Degano, Flemming Nielson, and Hanne Riis Nielson. Static validation of security protocols. *J. Comput. Secur.*, 13(3):347–390, 2005.
3. M. Boreale, R. Bruni, L. Caires, R. De Nicola, I. Lanese, M. Loreti, F. Martins, U. Montanari, A. Ravara, D. Sangiorgi, V. Vasconcelos, and G. Zavattaro. SCC: a Service Centered Calculus. In *Proc. of WS-FM 2006, 3rd Int. Workshop on Web Services and Formal Methods*, volume 4184 of *Lecture Notes in Computer Science*, pages 38–57. Springer Verlag.
4. W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *J. Log. Program.*, 1(3):267–284, 1984.
5. Jérôme Feret. Dependency analysis of mobile systems. In Daniel Le Métayer, editor, *ESOP*, volume 2305 of *Lecture Notes in Computer Science*, pages 314–330. Springer, 2002.
6. C. Guidi, R. Lucchi, R. Gorrieri, N. Busi, and G. Zavattaro. 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.
7. R. R. Hansen, C. W. Probst, and F. Nielson. Sandboxing in myklaim. In *Availability, Reliability and Security (ARES)*, pages 174–181. IEEE Computer Society, 2006.



8. 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.
9. R. Milner. The polyadic pi-calculus: a tutorial. In F. L. Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993.
10. R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.
11. F. Nielson, H. Riis Nielson, and C. L. Hankin. *Principles of Program Analysis*. Springer, 1999. Second printing, 2005.
12. F. Nielson, H. Riis Nielson, and H. Seidl. Cryptographic analysis in cubic time. *Electronic Notes of Theoretical Computer Science*, 62:7–23, 2002.
13. F. Nielson, H. Riis Nielson, H. Sun, M. Buchholtz, R. R. Hansen, H. Pilegaard, and H. Seidl. The Succinct Solver Suite. In K. Jensen and A. Podelski, editors, *TACAS*, volume 2988 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 2004.
14. 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.
15. F. Nielson, H. Seidl, and H. Riis Nielson. A succinct solver for ALFP. *Nord. J. Comput.*, 9(4):335–372, 2002.
16. Arnaud Venet. Automatic determination of communication topologies in mobile systems. In Giorgio Levi, editor, *SAS*, volume 1503 of *Lecture Notes in Computer Science*, pages 152–167. Springer, 1998.

## A Analysability and Subject Reduction

We have shown the correctness of our analysis in terms of a subject reduction result in Theorem 1. In order to obtain invariance under subject reduction, a correctness predicate needed to take care of the implicit substitution prevalent in standard reduction semantics. This is indeed necessary, because mere analysability is *not preserved* by subject reduction:

**Proposition 1.** *If  $P \rightarrow Q$  and  $\hat{R}, \hat{K} \vdash_{L,F} P$  then  $\hat{R}, \hat{K} \vdash_{L,F} Q$  does not hold necessarily.*

To see this consider the following excerpt from our running example presented in Table 1. The following computation step of the operational semantics is due to an application of rule (COM) describing the reception of a critical message by the black box process.

$$\dots \mid \text{status}(\text{car}_i; x)^4. [x = \text{crit}]^5. ! \text{alarm}(\text{car}_i)^6 \rightarrow [\text{crit} = \text{crit}]^5. ! \text{alarm}(\text{car}_i)^6$$

As shown above in Example 2, an acceptable analysis result  $(\hat{R}, \hat{K})$  for the process before the application of (COM) comprises  $\hat{R}.5 = \{\langle \text{ok} \rangle, \langle \text{crit} \rangle\}$  and  $\hat{R}.6 = \{\langle \text{crit} \rangle\}$ . However, we can now observe that this *cannot* be part of an acceptable analysis result for the derived process. The clause for matching, (RMATCH), will require  $\hat{R}.5 \subseteq \hat{R}.6$  since the test will hold for all the bindings of the variables of  $\hat{R}.5$ . But this does not hold and hence we cannot have a subject reduction result of the form suggested above.

The stronger correctness predicate of Definition 1 does hold for the derived process:

$$\hat{R}, \hat{K} \models^{P_{\text{acc}}} [\text{crit} = \text{crit}]^5. ! \text{alarm}(\text{car}_i)^6$$

This is proven by choosing  $\bar{w} \in \hat{R}.5$  of Definition 1 to be  $\langle \text{crit} \rangle$ .