LIX Colloquium 2006

Theoretical Aspects of Communication-Centred Programming[1]

Marco Carbone [2]

Department of Computing
Imperial College London
London, United Kingdom

Kohei Honda[3]

Department of Computer Science
Queen Mary and Westfield College
London, United Kingdom

Nobuko Yoshida[4]

Department of Computing
Imperial College London
London, United Kingdom

Abstract

This short note outlines two different ways of describing communication-centric software in the form of formal calculi and discuss their relationship. Two different paradigms of description, one centring on global message flows and another centring on local (end-point) behaviours, share the common feature, *structured representation of communications*. The global calculus originates from Web Services - Choreography Description Language (WS-CDL), a web service description language developed by W3C's WS-CDL Working Group. The local calculus is based on the π-calculus, one of the representative calculi for communicating processes. We illustrate these two descriptive frameworks, outline the static and dynamic semantics of these calculi, and discuss the basic idea of *end-point projection*, by which any well-formed description in the global calculus has a precise representation in the local calculus.

Keywords: Web Services, π-Calculus, Choreography, End-Point Projection, Session Types.

1 Introduction

The main purpose of this note is to introduce a formal calculus for communication and concurrency which fundamentally differs from existing concurrency formalisms.

[1] We would like to thank Gary Brown and Steve Ross-Talbot for their collaboration in the development of this work. This work is partially supported by EPSRC GR/R03075/01, GR/T04236/01, GR/S55538/01, GR/T04724/01, GR/T03208/01 and IST-2005-015905 MOBIUS.

[2] Email: carbonem@dcs.qmul.ac.uk

[3] Email: kohei@dcs.qmul.ac.uk

[4] Email: yoshida@doc.ic.ac.uk

This calculus is based on the idea of global description of interactions. It was born as a result of a dialogue between us and W3C's working group on web services standards, the Web Services Choreography Description Language Working Group (WS-CDL WG) where Robin Milner and the authors have been working as invited experts. This group was formed by W3C in 2003 for standardising a language for web services business protocols specification. Recognising the need for a foundational theory on which the design and infrastructure of the language is to be built on, the working group has taken a strong interest in the π-calculus. Robin Milner was originally invited alone and had offered many insights on the initial design of WS-CDL: for example, an implementation of fresh channels. He also encouraged the members to develop a formal calculus for WS-CDL. Honda and Yoshida were later invited under Milner's recommendation because of their knowledge on advanced typing systems for the π-calculus: WS-CDL needed a static validation of business protocols by type-checking to ensure deadlock-freedom and livelock-freedom. In 2005, Carbone joined the working group for developing formalisms and typing systems for a direct use for WS-CDL.

Related descriptive methods have been practiced in various contexts, including the standard notation for cryptographic protocols [11], message sequence charts (MSC) [10,8], a language related with MSC called DiCons [1], and UML sequence diagrams [12]. One may also view a Petri-net based description of various systems [14,2] as an instance of such global descriptions. The use of types in the formalism is the main difference from these notations. As its origin in W3C standardisation suggests, this formalism is distillation of engineering needs for describing complex interaction which may occur in real world business processes. The associated long version [6] presents extensive examples of business protocols written in the proposed global formalism.

A second aim of this note is to outline a theory which rigourously relates this design-oriented formalism to a typed process calculus, which is based on the notion of processes and their interaction. Establishing this connection is important since, through this link, a rich theory of algebras, calculi and logics for processes becomes available for direct dialogue with engineering practice. The target of the mapping is an applied version of the π-calculus with locations based on session types. The mapping from the global formalism to the process formalism is called *end-point projection*, or EPP for short, following the terminology of WS-CDL (where EPP is regarded as an underpinning of a variety of web service engineering, including monitoring, conformance and interoperatbility). EPP projects a given global description to a collection of end-point behaviours, whose mutual communication should realise the original global scenario. We naturally desire EPP to be *sound* and *complete*, in the sense that all and only globally described behaviour is realised in interactions among end-points. To make this possible, we impose simple descriptive principles to global descriptions. For those global descriptions which are well-typed and which follow these descriptive principles, there is a simple and direct EPP which is type preserving and which is sound and complete with respect to the original dynamic semantics. This EPP theory (including type disciplines and EPP mapping) will be published as a supplementary document to the specification of WS-CDL 1.0, and will be implemented as part of an open-source reference implementation of WS-

CDL [13]. Thus, the present work started from practice, gets related to theory, and finally again comes back to practice. This will further motivate and encourage a dialogue between theoretical studies and engineering. We argue such a dialogue is highly beneficial, not only for advancement of rigourous engineering but also for enrichment of theory.

It is worth outlining the direct engineering background of the present work, WS-CDL. WS-CDL has been developed in order to meet engineering needs for the development of business protocols on the world-wide web. The central engineering idea of WS-CDL is embodied by the term *choreography* in its name. The underlying intuition can be summarised as follows:

"Dancers dance following a global scenario without a single point of control."

WS-CDL is about writing down such a "global scenario": in computational terms, it is a language for describing business protocols from a global viewpoint such that the description can be executed by individual distributed processes without a single point of control. In other words, if a designer writes a global description in WS-CDL, it should be realisable as communicating processes without any central controlling agent (which contrasts with the notion of *orchestration*, where one master component, "conductor", directly controls the activities of one or more slave components). Thus the notion of choreography intrinsically *demands* an appropriate framework of EPP.

A broader background of the present work is the explosive growth of the Internet and world-wide web which has given rise to, in the shape of de facto standards, an omnipresent naming scheme (URI/URL), an omnipresent communication protocols (HTTP/TCP/IP) and an omnipresent data format (XML). These three elements arguably offer the key infra-structural bases for application-level distributed programming. This engineering background makes it feasible and advantageous to develop applications which will be engaged in complex sequences of interactions among two or more parties. Another background is maturing out of theories of processes centring on the π-calculus and its types. The two formalisms introduced in this note are based on a common notion of structured communication, called *session*. A session binds a series of communications between two parties into one, distinguishing them from communications belonging to other sessions. This is a standard practice in business protocols (where an instance of a protocol should be distinguished from another instance of the same or other protocols) and in distributed programming (where two interacting parties use multiple TCP connections for performing a unit of conversation). The type disciplines for sessions have been studied over long years in the context of the π-calculus [9,7,15,3], where it has been shown that they offer a high-level of abstraction for communication behaviour upon which further refined reasoning techniques can be built. Not only sessions offer a natural articulation for a global description of complex interactions but they also play an essential role in the presented theory of end-point projection.

The next section of this abstract briefly outlines the global formalism (a distilled version of WS-CDL), the corresponding process formalism (called *end-point calculus*), and a theory of EPP. A prior paper [4] illustrates the use of the global calculus for describing communication behaviour through examples; whereas [5] presents an associated theory. Exploration of many examples, the full technical development of the theory, and detailed discussions on related works, can be found in [6], a contribution by the whole of the π-calculus experts team to the aforementioned W3C standard.

2 Outline of Calculi and Formal Results

The formal syntax of the global calculus is given by standard BNF. Below symbols $I, I', \ldots$ denote *terms* of the global calculus, also called *interactions*. Terms describe a course of information exchange among two ore more parties from a global viewpoint. Moreover, $a, b, c, ch, \ldots$ range over *service channels*, which may be considered as shared channels of web services; $s, s', \ldots$ range over *session channels*, which designate communication channels freshly generated for each session; $\tilde{s}$ indicates a their vector; $A, B, C, \ldots$ range over *participants*. Each participant is equipped with its own local state, storing and updating values in its variables $(x, y, z, \ldots)$; $X, Y, Z, \ldots$ range over *term variables*, which are used to represent recurrence in combination with recursion $\mathbf{rec}\ X.I$; and a $e, e', \ldots$ range over arithmetic and other standard first-order expressions.

$$
\begin{array}{rll}
I ::= & A \to B : c(\boldsymbol{\nu}\,\tilde{s})\,.\,I & \text{(init)} \\
 & \mid A \to B : s\langle \mathsf{op},\ e,\ y\rangle\,.\,I & \text{(com)} \\
 & \mid x@A := e\,.\,I & \text{(assign)} \\
 & \mid I_1 \mid I_2 & \text{(par)} \\
 & \mid \mathsf{if}\ e@A\ \mathsf{then}\ I_1\ \mathsf{else}\ I_2 & \text{(ifthenelse)} \\
 & \mid I_1 + I_2 & \text{(sum)} \\
 & \mid (\boldsymbol{\nu} s)\ I & \text{(new)} \\
 & \mid X & \text{(recVar)} \\
 & \mid \mathbf{rec}\ X\,.\,I & \text{(rec)} \\
 & \mid \mathbf{0} & \text{(inaction)}
\end{array}
$$

We briefly comment the various operations. The operation (init) is about session initiation and states that participant A opens a session with participant B on service channel ch with session channels $\tilde{s}$ and then continue as I. Communication of values is instead specified by (com) where A sends over session channel s with operation op the value e to B which will store such a value at variable y. (assign) is the standard assignment i.e. the variable x located at A is updated with e. (com) and (sum) are respectively the parallel composition and the non-deterministic choice of interactions. (ifthenelse) is the standard conditional operation. while (recVar) and (rec) are used to express recursive behaviour of interactions. $\mathbf{0}$ is the inactive interaction.

The following example, the *Buyer-Seller protocol* from the WS-CDL WG, depicts a repeated interaction where Buyer asks Seller to give a good quote. Only when a quote is good, Buyer accepts, and Seller sends a shipping request to Shipper.

$$
\begin{array}{l}
\textbf{Buyer} \to \textbf{Seller} : \mathtt{ch1}(\boldsymbol{\nu}\, s, t).\\
\textbf{Buyer} \to \textbf{Seller} : s\langle \mathsf{QuoteReq},\ \mathit{prod}@\text{Buyer},\ \mathit{prod}@\text{Seller}\rangle.\\
\textbf{rec}\ X.\ \textbf{Seller} \to \textbf{Buyer} : t\langle \mathsf{QuoteRes},\ \mathit{quote}@\text{Seller},\ \mathit{quote}@\text{Buyer}\rangle\ .\\
\quad \textbf{if}\ \mathsf{reasonable}(\mathit{quote})@\text{Buyer}\ \textbf{then}\\
\quad\quad \textbf{Buyer} \to \textbf{Seller} : s\langle \mathsf{QuoteAcc},\ \mathit{adr}@\text{Buyer},\ \mathit{adr}@\text{Seller}\rangle.\\
\quad\quad \textbf{Seller} \to \textbf{Shipper} : \mathtt{ch2}(\boldsymbol{\nu}\, r).\\
\quad\quad\quad \text{...(omitted)...}\\
\quad \textbf{else}\\
\quad\quad\quad \textbf{Buyer} \to \textbf{Seller} : s\langle \mathsf{QuoteNoGood}\rangle.X
\end{array}
$$

where, for example, "*prod*@Buyer" indicates a variable *prod* located at Buyer. Many other examples of descriptions of complex protocols can be found in [6].

The global calculus is equipped with a formal semantics in terms of reduction and it is defined using an intuitive notation,

$$(I,\ \sigma)\ \to\ (I',\ \sigma')$$

which says a global description I in a state σ (which is the collection of all local states of the participants) will reduce to I' in a new configuration σ'. Samples of reduction rules are, writing $\sigma[x@B \mapsto v]$ for the result of updating the variable x located at B:

$$
\begin{array}{lr}
(A \to B : \mathtt{ch}(\boldsymbol{\nu}\, \tilde{s}).\, I,\ \sigma)\ \to\ ((\boldsymbol{\nu}\, \tilde{s})I,\ \sigma) & \\
(A \to B : s\langle \mathsf{op},\ v,\ x\rangle.\, I,\ \sigma)\ \to\ (I,\ \sigma[x@B \mapsto v]) & \\
(x@A := v.\, I,\ \sigma)\ \to\ (I,\ \sigma[x@A \mapsto v]) & \\
(I_1 + I_2,\ \sigma)\ \to\ (I_1',\ \sigma') & \text{if } (I_1, \sigma)\ \to\ (I_1', \sigma')\\
(I_1 \mid I_2,\ \sigma)\ \to\ (I_1' \mid I_2,\ \sigma') & \text{if } (I_1, \sigma)\ \to\ (I_1', \sigma')
\end{array}
$$

Note updates of stores in the second and third rules are local to designated participants. Also note that there is no synchronisation of parallel interactions: communications are specified only by single interactions (global view).

As mentioned above, the type discipline for the calculus is based on session types. Writing $\theta, \theta_i, ...$ for first-order value types, the grammar of types is given as:

$$
\begin{array}{lll}
\alpha & ::= & s \blacktriangleright \Sigma_i \langle \mathsf{op_i}, \theta_i\rangle.\ \alpha_i \quad | \quad s \blacktriangleleft \Sigma_i \langle \mathsf{op_i}, \theta_i\rangle.\ \alpha_i \quad | \quad \alpha_1 \,|\, \alpha_2 \quad | \quad \textbf{rec}\ t.\alpha\\
 & | & t \quad | \quad \mathbf{0}
\end{array}
$$

while the typing sequent has the form: $\Gamma\ \vdash\ I\ \triangleright\ \Delta$ where:

- Γ typically contains a type assignment of the form $\mathtt{ch}@A : (\tilde{s})\alpha$, which says a service channel $\mathtt{ch}$ at A may be invoked with fresh $\tilde{s}$ followed by a session α.
- Δ typically contains a type assignment of the form $\tilde{s}[A, B] : \alpha$, which says a session of type α from A to B takes place via $\tilde{s}$ (where α uses at most $\tilde{s}$).

We omit the typing rules for which we have the standard properties such as subject reduction and minimal (principal) typing. See [6] for further details.

We now introduce the end-point calculus. The following grammar defines *processes* $(P, Q, \ldots)$ and *networks* $(M, N, \ldots)$.

$$\begin{array}{rl} P ::= & \mathbf{!}\,\mathsf{ch}(\tilde{s}).P \quad | \quad \overline{\mathsf{ch}}(\boldsymbol{\nu}\,\tilde{s}).P \quad | \quad s \rhd \Sigma_i \mathsf{op}_i(\tilde{y}_i).P_i \quad | \quad \overline{s} \lhd \mathsf{op}\langle\tilde{e}\rangle.P \\ & |\; x := e.P \quad | \quad \mathsf{if}\ e\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \quad | \quad P \oplus Q \quad | \quad P \,|\, Q \quad | \quad (\boldsymbol{\nu}\, s)P \\ & |\; X \quad | \quad \mathbf{rec}\ X.P \quad | \quad \mathbf{0} \end{array}$$

$$N ::= \quad A[P]_\sigma \quad | \quad N_1 \,|\, N_2 \quad | \quad (\boldsymbol{\nu}\, s)N \quad | \quad \boldsymbol{\epsilon}$$

where $A[P]_\sigma$ indicates a participant A whose behaviour is given by P and whose local state is σ. As in [9], we have two operations for session initiation where the input $\mathbf{!}\,\mathsf{ch}(\tilde{s}).P$ is replicated i.e. always available after invocation. The operation $s \rhd \Sigma_i \mathsf{op}_i(\tilde{y}_i).P_i$ is the branching input where each choice is expressed by a different op_i. The dual operation is the output $\overline{s} \lhd \mathsf{op}\langle\tilde{e}\rangle.P$. The other operations are standard.

Reduction is given modulo the standard structural equality, with sample rules:

$$\begin{array}{l} A[\mathbf{!}\,\mathsf{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \;\;|\;\; B[\overline{\mathsf{ch}}(\boldsymbol{\nu}\,\tilde{s}).Q \,|\, S]_{\sigma_B} \\ \qquad \rightarrow \;(\boldsymbol{\nu}\,\tilde{s})(A[P \,|\, \mathbf{!}\,\mathsf{ch}(\tilde{s}).P \,|\, R]_{\sigma_A} \;\;|\;\; B[Q \,|\, S]_{\sigma_B}) \end{array}$$

$$\begin{array}{l} A[s \rhd \Sigma_i \mathsf{op}_i(y_i).P_i \,|\, R]_{\sigma_A} \;\;|\;\; B[\overline{s} \lhd \mathsf{op_j}\langle v\rangle Q \,|\, S]_{\sigma_B} \\ \qquad \rightarrow \;A[P_j \,|\, R]_{\sigma_A[y_j \mapsto v]} \;\;|\;\; B[Q \,|\, S]_{\sigma_B} \end{array}$$

The typing for the end-point calculus uses the same set of types, with the two forms of sequent, one for processes and one for networks (Γ and Δ are as above):

$$\Gamma \vdash_A P \rhd \Delta, \qquad\qquad \Gamma \vdash M \rhd \Delta$$

where $\Gamma \vdash_A P \rhd \Delta$ designates, as subscript, the participant A in which the subject process P is to be located. The typing rules are those of the standard session typing [9] (extended to multiple session channels). One basic rule is the following initialisation rule:

$$(\textsc{Init})\; \frac{\Gamma,\, ch@A\!:\!(\tilde{s})\alpha \;\vdash_A\; P \;\rhd\; \tilde{s}\!:\!\alpha}{\Gamma,\, ch@A\!:\!(\tilde{s})\alpha \;\vdash_A\; \mathbf{!}\,ch(\tilde{s}).P \;\rhd\; \emptyset}$$

which demands linear session channels in the premise are abstracted in the conclusion (note the service channel ch is replicated). The typing system satisfies the standard subject reduction, freedom from type errors, and the minimal (principal) typing.

Finally we translate a global description into its end-point counterpart (end-point projection, or EPP). The process of EPP can however be tricky, because we can easily produce a global description which does not have a local counterpart. We have identified three descriptive principles [6], which are:

- *Connectedness*, a basic local causality principle.
- *Well-threadedness*, a stronger locality principle based on session types.
- *Coherence*, a consistency principle for description of each participant.

All these principles are stipulated incrementally on the basis of well-typedness. They not only enunciate natural disciplines for well-structured global descriptions, but also offer gradually deeper analysis of operational aspects, guiding us to the simple definition of an EPP map of essentially the following shape:

$$(I,\ \sigma) \quad \mapsto \quad A[P]_{\sigma@A} \ \mid\ B[Q]_{\sigma@B} \ \mid\ C[R]_{\sigma@C} \ \mid\ \cdots$$

where P is the projection of I onto A, similarly for others. $\sigma@A$ projects σ onto A. In [6] we have established that, when applied to well-structured interactions, the EPP mapping thus defined satisfies the following three properties:

- *Type preservation:* the typing is preserved through EPP.
- *Soundness:* nothing but behaviours (reductions) in I are in the image of its EPP.
- *Completeness:* all behaviours (reductions) in I are in the image of its EPP.

Thus the resulting processes/networks never have a type error, and they realise all and only interactions prescribed in the original global description.

We conclude this note with a small example showing how EPP works. We consider the Buyer-Seller example seen in page 4 and give its end-point code. The participants involved are Buyer, Seller and Shipper, hence we need to generate the end-point code for each of them. We start from the code for Buyer:

$$\begin{array}{l}
\overline{\texttt{ch1}}(\boldsymbol{\nu}\, s,t)\,.\,\overline{s} \lhd \mathsf{QuoteReq}\langle prod@\text{Buyer}\rangle\,.\\
\textbf{rec}\ X\,.\,t \rhd \mathsf{QuoteRes}(quote@\text{Buyer})\,.\\
\mathsf{if}\ \texttt{reasonable}(quote)@\text{Buyer}\ \mathsf{then}\\
\qquad \overline{s} \lhd \mathsf{QuoteAcc}\langle adr@\text{Buyer}\rangle\,.\ \ ...\text{(omitted)}...\\
\mathsf{else}\\
\qquad \overline{s} \lhd \mathsf{QuoteNoGood}\langle\rangle\,.\,X
\end{array}$$

Note that the conditional construct is implemented by Buyer as that is where the guard is evaluated. On the other hand, Seller behaves as

$$\begin{array}{l}
!\texttt{ch1}(s,t)\,.\,s \rhd \mathsf{QuoteReq}\langle prod@\text{Seller}\rangle\,.\\
\textbf{rec}\ X\,.\,\overline{t} \lhd \mathsf{QuoteRes}(quote@\text{Seller})\,.\\
(\ s \rhd \mathsf{QuoteAcc}\langle adr@\text{Seller}\rangle\,.\,\overline{\texttt{ch2}}(\boldsymbol{\nu}\, r)\,.\ \ ...\text{(omitted)}...\\
\quad +\\
\ \ s \rhd \mathsf{QuoteNoGood}()\,.\,X\ )
\end{array}$$

On this side, the conditional construct does not appear. In fact, Seller must be independent from whatever the choice at Buyer is. This is realised by the external choice where both operations, QuoteAcc and QuoteNoGood are offered on session channel s.

The behaviour of Shipper is defined similarly. We notice that there must be a replicated process in the form of $!\mathtt{ch2}(r)\,\mathbf{.}\,P$, which offers the service requested by Seller.

References

[1] Baeten, J., H. van Beek and S. Mauw, *Specifying internet applications with DiCons*, in: *SAC'01* (2001), pp. 576–584.

[2] Basten, T., "In Terms of Nets," Ph.D. thesis, Eindhoven University of Technology (1998).

[3] Bonelli, E., A. B. Compagnoni and E. L. Gunter, *Correspondence assertions for process synchronization in concurrent communications.*, JFP **15** (2005), pp. 219–247.

[4] Carbone, M., K. Honda and N. Yoshida, *A calculus of global interaction based on session types*, in: *2nd Workshop on Developments in Computational Models (DCM)*, ENTCS, 2006, submitted.

[5] Carbone, M., K. Honda and N. Yoshida, *Structured communication-centred programming for web services*, in: *Proc. of ESOP'07*, LNCS, 2007.

[6] Carbone, M., K. Honda, N. Yoshida, R. Milner, G. Brown and S. Ross-Talbot, *A theoretical basis of communication-centred concurrent programming*, To be published by W3C. Available at http://www.dcs.qmul.ac.uk/∼carbonem/cdlpaper/workingnote.pdf (2006).

[7] Dezani-Ciancaglini, M., D. Mostrous, N. Yoshida and S. Drossopoulou, *Session Types for Object-Oriented Languages*, in: *Proceedings of ECOOP'06*, LNCS, 2006.

[8] Foster, H., "A Rigorous Approach To Engineering Web Service Compositions," Ph.D. thesis, Imperial College London, University Of London, Department of Computing (2006).

[9] Honda, K., V. Vasconcelos and M. Kubo, *Language primitives and type disciplines for structured communication-based programming*, in: *ESOP'98*, LNCS **1381**, 1998, pp. 22–138.

[10] International Telecommunication Union, *Recommendation Z.120: Message sequence chart* (1996).

[11] Needham, R. M. and M. D. Schroeder, *Using encryption for authentication in large networks of computers.*, Commun. ACM **21** (1978), pp. 993–999.

[12] OMG, *Unified modelling language, version 2.0* (2004).

[13] PI4SOA, *http://www.pi4soa.org.*

[14] van der Aalst, W., *Inheritance of interorganizational workflows: How to agree to disagree without loosing control?*, Info. Tech. and Management J. **2** (2002), pp. 195–231.

[15] Vasconcelos, V., A. Ravara and S. J. Gay, *Session types for functional multithreading.*, in: *CONCUR'04*, LNCS **3170**, 2004, pp. 497–511.