Submitted to the Technical Communications of the International Conference on Logic Programming (ICLP'10)
http://www.floc-conference.org/ICLP-home.html

MODELS FOR TRUSTWORTHY SERVICE AND PROCESS ORIENTED SYSTEMS

HUGO A. LÓPEZ [1]

Programming, Logic and Semantics group, IT University.
Rued Langgaards Vej 7, 2300 Copenhagen, Denmark
E-mail address: lopez@itu.dk
URL: http://www.itu.dk/~hual/

Abstract. Service and process-oriented systems promise to provide more effective business and work processes and more flexible and adaptable enterprise IT systems. However, the technologies and standards are still young and unstable, making research in their theoretical foundations increasingly important. Our studies focus on two dichotomies: the global/local views of service interactions, and their imperative/declarative specification. A global view of service interactions describes a process as a protocol for interactions, as e.g. an UML sequence diagram or a WS-CDL choreography. A local view describes the system as a set of processes, e.g. specified as a π -calculus or WS-BPEL process, implementing each participant in the process. While the global view is what is usually provided as specification, the local view is a necessary step towards a distributed implementation. If processes are defined imperatively, the control flow is defined explicitly, e.g. as a sequence or flow graph of interactions/commands. In a declarative approach processes are described as a collection of conditions they should fulfill in order to be considered correct. The two approaches have evolved rather independently from each other. Our thesis is that we can provide a theoretical framework based on typed concurrent process and concurrent constraint calculi for the specification, analysis and verification of service and process oriented system designs which bridges the global and local view and combines the imperative and declarative specification approaches, and can be employed to increase the trust in the developed systems. This article describes our main motivations, results and future research directions.

1. Introduction

As recently pointed out by the ICT theme of EU Seventh Framework Programme (FP7), the need of trustworthy and pervasive services infrastructure is considered one of the three mayor challenges in ICT for the next ten years. The "future internet" proposes questions in terms of scalability, mobility, flexibility, security, trust and robustness to the more than thirty years old current Internet architecture. A vast landscape of application and ever-changing requirements and environments must be supported, and new ways of interaction must be devised, coping with safety and reliability in their coordination methods.

1998 ACM Subject Classification: F.3.2: Semantics of Programming Languages, F.4.3: Formal Languages.

Key words and phrases: Concurrent Constraint Calculi, Session Types, Logic, Service and Process oriented computing.

The line of research investigating such questions has been constantly expanding since the early nineties, both combining approaches from the academia and the industry. As result of such efforts, its been normally hard to differentiate between similar derived fields, like Business Processes, Workflow technologies and Service Oriented Computing. A Business Process is the set of steps executed in order to fulfill a (business) goal. Business processes have always been at the hearth of companies interests, and the obvious goal has been to develop better, cheaper, and faster processes, incrementing the profits of the company. Workflows came as an initial response for the need of proper descriptions of business processes, providing a framework for the specification and automation of processes by means of activities respecting a business logic. They aim at integrating coarse-grained components and have a single place where the business logic is specified. Furthermore, Service Oriented Computing (SOC) opens a new different horizon by distributing the places where the business logic is defined: now, small process units (services) can be shared between different organizations, so each of them can fulfill their business goals by reusing and outsourcing services.

Giving the intrinsic complexity when analyzing services in distributed environments, one normally use different abstractions to describe and analyse services. One of such abstractions deals with the the study of the *concurrent* nature of services. *Process calculi* are formal languages conceived for the description and analysis of concurrent systems. As such, the goal of a process calculus is to provide a rigorous framework where complex systems can be accurately analyzed, including reasoning techniques (type systems, specification logics) to verify essential properties of a system. The term *structured communications* [9] refers to the branch of process calculi devoted to the analysis of interactions between services. On a calculus for structured communications, one considers the computation within a service as an atomic activity, and focus the core of the analysis in the interactions between services.

One of the most important aspects when modelling services relate to the notion of *trustworthiness*, or the extent to which users believe that the systems behave correctly. A *safe* system is one in which a property considered harmful for the life of the system would never happen, like for instance the disclosure of the private credentials of the manager to a thief.

Despite of being such a young trend, different but interrelated views for the analysis of service oriented systems have been proposed. We can enclose such approaches in two dichotomies: *global/local* views of services, and *imperative/declarative* specifications. In the first dichotomy, either one describe the system as the exchange of messages between different participants, or one consider the system as the composition of the local behaviours of each participant. In this first view, known as *choreography* [4, 5], one consider the system as a whole, taking care only of the interfaces that participants use when interacting to the outside world. In the second view, known as *orchestration* [14, 5], one model the system as perceived by the eyes of each participant (so-called *end-point*), sending and receiving messages but not knowing which other actors are present in a communication. As recently presented, choreographies and orchestrations can be operationally correspondent, and one can either project a choreography to generate distributed orchestrations that implements it, or lift a process specification done in an orchestrated manner to describe its respective choreography [5].

As a simple example of such duality, take a simplified version of an online booking scenario: Here, the customer interacts with the airline company AC using its service *ob*, such interaction will be labelled by an identifier (referred here as a *session*). The customer and

AC can interact in more than one manner, requiring sessions to be unique and independent from each other. In this case, we will use sessions labelled k_1, k_2 to identify the direction of the communication: k_1 from Customer to the AC and k_2 for its dual. Once sessions are established, the customer will request the company about a flight offer with his booking data, along the session key k_1. The airline company will process the customer request and will send a reply back with an offer using the session key k_2. The customer will eventually accept the offer, sending back an acknowledgment to the airline company using k_1. The description of this protocol in a choreographic way will describe the sequence of interactions between Customer and AC, for instance: $Customer \longrightarrow AC : ob(k_1, k_2)$ will create sessions k_1 and k_2 between Customer and AC, and $Customer \longrightarrow AC : k_1\langle booking, x\rangle$ will describe the communication of the *booking* value using the session key k_1 from Customer to AC. The rest of the specification representing this protocol can be described as follows:

$$
\begin{aligned}
&Customer \longrightarrow AC : ob(k_1, k_2).\\
&Customer \longrightarrow AC : k_1\langle booking, x\rangle.\\
&AC \longrightarrow Customer : k_2\langle offer, y\rangle.\\
&Customer \longrightarrow AC : k_1\langle accept, z\rangle
\end{aligned}
$$

In an orchestrated version of the above example, one might consider the system as the concurrent execution of processes implementing the actions for Customer and AC. Here, processes will communicate via *session establishment* and *message passing*, among other actions. Following the notation from [9], the concurrent execution of **request** $ob(k)$ **in** P and **accept** $ob(k)$ **in** P will create a session k between P and Q, whereas $k![booking]; P$ in parallel with $k?(x)$ **in** Q will use a previously established session k to communicate the data contained in *booking* from P to Q. The specification of the example is coded below, using $\|$ to denote parallel composition of processes, $(\nu x)\ P$ as the creation of a new resource x local to P, and $\mathbf{0}$ the termination of a process:

$$
\begin{aligned}
Customer &= \textbf{request}\ ob(k_1, k_2)\ \textbf{in}\ (k_1![booking]; k_2?(y)\ \textbf{in}\ (k_1![accept]; \mathbf{0}))\\
AC &= \textbf{accept}\ ob(k_1, k_2)\ \textbf{in}\ (k_1?(x)\ \textbf{in}\ (\nu\ offer)k_2![offer]; k_1?(z)\ \textbf{in}\ \mathbf{0})\\
System &= Customer \parallel AC
\end{aligned}
$$

Here, the communication will be *structured* if we can provide guarantees about the use of sessions along the life of the protocol. For instance, considering the choreographic specification of the example given above, we can guarantee that the usage of sessions will require first an interaction using k_1, followed by k_2 and finalized by k_1. It is obvious that such guarantees become harder to express in architectures with thousands of services, which is the case of service oriented architectures.

The second dichotomy here considered refers to the approach used to construct the models. Descriptions can have imperative or declarative flavors: In an imperative approach, one explicitly define the control flow of commands. Typical representatives of this approach are based on process calculi, and come with behavioral equivalences and type disciplines as their main analytic tools [18, 10, 2, 9, 21]. On the contrary, in a declarative approach the

focus drifts to the specification of the set of constraints (causality relations, time constraints, quality of service) processes should fulfill in order to be considered correct [17, 20, 12, 15]. Even if these two trends address similar concerns, we find that they have evolved rather independently from each other. Returning to our example, we might consider the specifications above presented imperative specifications, whereas a declarative specification will let parts of the process unspecified. For instance, we could relax the specification given above by accepting any implementation of AC that complies with an ordering of actions where it first receives the booking data, and eventually (that is, immediately or in an unspecified sequence of interactions) returns a booking offer. Such a policy can be observed better on a logical formalization, as for instance a formula in Linear Temporal Logic [13].

2. A unifying framework for structured communications

This research has as a main objective to leverage the trustworthiness level of a system by providing a clear methodology of specification and verification of structured communications. Our goal is to give characterizations of services, both at the *operational* and *logical* level. This is done by relating the way services are specified, both from their global and local viewpoints. Figure 1 illustrates the approach for the specification and verification of structured communications. A specification of a choreography C can be projected to the parallel composition of end points P_i with an index i corresponding to each of the participants involved in choreography. Similarly, every choreographic specification in C corresponds to a formula in a modal logic representing the interactions between agents; such a correspondence is described in the figure as the bijection $\mathcal{GL}$ between C and ϕ_C. $\mathcal{GL}$ not only provides a logical characterization of a process; it also allows for *partial specification*: Given a logical formula, one can see if there is a process in C that can satisfy ϕ_C.

A similar reasoning is provided for orchestrations: Starting with i-indexed parallel composition implementing each participant P_i (denoted $\prod_i [\![P_i]\!]$), one is interested in describing the behaviour of its composition. Such description is embedded in the bijection $\mathcal{LL}$ between the orchestration in $\prod_i [\![P_i]\!]$ and its logical counterpart in $\bigcap_i [\![\chi_i]\!]$. Moreover, a formula representing the global behaviour of a choreography can be projected to a corresponding formula describing the behaviour of a set of orchestrations. Such a mapping can be observed in the diagram as the function LP from ϕ_C to $\bigcap_i [\![\chi_i]\!]$.

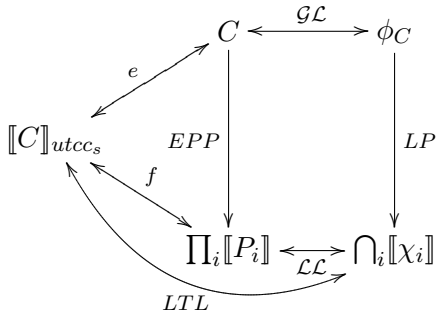

Figure 1: Methodology for the verification of structured communcations

Finally, one can observe an interesting relation when comparing languages of structured communications and other models of concurrency. CC refers to the Concurrent Constraint

family of languages [19] and its timed extensions, such as timed CC (tcc) [6] and universal tcc (utcc) [16]. We can see CC languages as part of the declarative approaches for the analysis of choreographies: First, differently from the classical approach where a value is assigned to each system variable *(store-as-valuation)*, in CC languages the store represents a *constraint* on the possible values of variables at one point in the life of the system. Second, it allows one to consider both the declarative flavour of logics and the execution of processes both in a single framework: the satisfaction of a formula allows the system to proceed, and the execution/inhibition of a process in the interaction is only defined by the amount of information available in the store. Timed extensions of the CC family refine the notion of store-as-constraint, describing the system as sequences of input-output stimuli between a set of processes and a store. These extensions give us enough modelling power to express declarative and imperative information in the same framework. The encodings between a choreography model and a timed CC specification are depicted by the function e, the corresponding mapping between a timed CC model and an orchestration model are depicted by the bijective function f; finally, the correspondence between an CC model and its logical counterpart is given by means of Linear Temporal Logic (LTL) [13].

3. Overview of Completed and Current Work

The evolution of this research project can be divided in three mayor research areas: First, we started by equipping CC languages with primitives for the analysis of structured communications, namely the treatment of mobile data and access control of information flow. A recent addition to the familiy of CC languages, known as Universal Timed CCP (`utcc`) introduces the possibility of universally quantify over predicates in the constraint store. `utcc` is presented as a candidate for representing mobility and security, both important concepts when talking about structured communications. However, the universal quantification in `utcc` is completely unrestricted. This means that in the proposed representations of link mobility and security protocols in `utcc`, every agent may guess channel names and encrypted values by universal quantification. It is thus necessary to enforce a restriction on the allowed processes to make sure that this is not possible. We proposed $\texttt{utcc}_s$, an extension of universal `tcc` with a type system for constraints used as patterns in process abstractions, which essentially allows us to distinguish between universally abstractable information and secure (non-leakable information) in predicates. We also proposed a novel notion of abstraction under local knowledge, which gives a general way to model that a process (principal) knows a key and can use it to decrypt a message encrypted with this key without revealing the key [8].

Second, we related CC and orchestration languages. We exploited `utcc` to give a declarative interpretation to the language of orchestrations at [9]. This way, services can be analyzed in a declarative framework where time is defined explicitly, and their behaviour compared to formulae in LTL. We do so by giving an operationally correspondent encoding of the language in [9] into `utcc`. Moreover, the selected language is prone to timed extensions: as we show in [11] an orchestration language can be benefitted from the inclussion of timed information on the duration of sessions, declarative preconditions within session establishment constructs, and session abortion primitives.

Finally, we filled the gap between choreographic models and logical specifications. Starting with an extension of Hennesy-Milner logic [7], we introduced $\mathcal{GL}$, a global logic for the

study of choreographies. $\mathcal{GL}$ describes properties over the transitions of a given choreography. As for structured communications, $\mathcal{GL}$ places special on the main elements of interactions in the choreography, namely the participants involved in a communication, the sessions used in an interaction and the effects on the variables by a given communication. The logic is equipped with a proof system that allows for verification of properties among participants in a choreography. With $\mathcal{GL}$, one can see the state of a choreography as a formula in the logic, and one can check for satisfaction of desirable properties by relating a logical formula wrt a choreographic specification [3].

4. Open Issues

The research being done to the moment constitutes just seminal steps on the path towards a verification framework of structured commmunications. Our main concerns relate to establishing a relation between the model of end-points and logical frameworks for the specification of sessions. In [1], Berger et al. presented proof systems characterizing May/Must testing preorders and bisimilarities over typed π -calculus processes. The connection between types and logics in such system comes in handy to restrict the shape of the processes one might be interested. In particular, being the synchronization methods in orchestration languages of similar nature as the ones present in the π calculus, one might consider such work as a suitable proof system for the calculus of end points. Our next step will focus on relating $\mathcal{GL}$ to orchestrations, both by providing a corresponding logic for the analysis of orchestration and by making the logical projection between global and local formulae. If successful, this research will contribute by providing a basis for logical specifications and model checking of structured communications. Finally, we want to continue the research on representing both global and local process views in concurrent constraint calculi, aiming at a unified representation of both views within the same formal model.

Acknowledgments

This research owes much to Thomas Hildebrandt for his indispensable guidance, and to Marco Carbone for the many insightful discussions profiling this topic of research. The research has been partially supported by the Computer Supported Mobile Adaptive Business Processes (www.CosmoBiz.org) project and the Trustworthy Pervasive Healthcare Services (www.Trustcare.dk) project, supported by the Danish Research Agency (grant no.: 274-06-0415 and grant no.: 2106-07-0019).

References

[1] M. Berger, K. Honda, and N. Yoshida. Completeness and logical full abstraction in modal logics for typed mobile processes. In L. Aceto, editor, *ICALP'08*, number 5126 in LNCS, pages 99–111. Springer-Verlag, Berlin Germany, 2008.

[2] M. Boreale, R. Bruni, L. Caires, R. De Nicola, I. Lanese, M. Loreti, F. Martins, U. Montanari, A. Ravara, and D. Sangiorgi. SCC: a service centered calculus. *Proceedings of WS-FM*, 4184:38–57, 2006.

[3] M. Carbone, T. Hildebrandt, and H. A. López. Towards a modal logic for the global calculus. In K. Honda and A. Mycroft, editors, *Programming Language Approaches to Concurrency and Communication-cEntric Software (PLACES)*, 2010.

[4] M. Carbone, 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.

[5] M. Carbone, K. Honda, and N. Yoshida. Structured communication-centred programming for web services. In *16th European Symposium on Programming (ESOP)*, volume 4421 of *LNCS*, pages 2–17, Braga, Portugal, March 2007. Springer, Berlin Heidelberg.

[6] F. de Boer, M. Gabbrielli, and M. Meo. A Timed Concurrent Constraint Language. *Information and Computation*, 161(1):45–83, 2000.

[7] M. Hennessy and R. Milner. On Observing Nondeterminism and Concurrency. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, pages 299–309. Springer-Verlag London, UK, 1980.

[8] T. Hildebrandt and H. A. López. Types for Secure Pattern Matching with Local Knowledge in Universal Concurrent Constraint Programming . In *International Conference on Logic Programming (ICLP)*, volume 5649 of *Lecture Notes in Computer Science*, pages 417–431. Springer, Berlin Heidelberg, 2009.

[9] K. Honda, V. Vasconcelos, and M. Kubo. Language Primitives and Type Discipline for Structured Communication-Based Programming. In *7th European Symposium on Programming (ESOP): Programming Languages and Systems*, pages 122–138. Springer-Verlag London, UK, 1998.

[10] A. Lapadula, R. Pugliese, and F. Tiezzi. A calculus for orchestration of web services. In *Proc. of 16th European Symposium on Programming (ESOP'07)*, volume 4421 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2007.

[11] H. A. López, C. Olarte, and J. A. Pérez. Towards a Unified Framework for Declarative Structured Communications. In *Programming Language Approaches to Concurrency and Communication-cEntric Software (PLACES'2009)*, volume 17 of *EPTCS*, pages 1–15, 2010.

[12] K. M. Lyng, T. Hildebrandt, and R. R. Mukkamala. The Resultmaker Online Consultant: From Declarative Workflow Management in Practice to LTL. In *Proc. of 1st Intl. Workshop on Dynamic and Declarative Business Processes (DDBP)*, Munich, Germany, 2008.

[13] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems: Specification*. Springer, 1992.

[14] J. Misra and W. R. Cook. Computation orchestration: A basis for wide-area computing. *Journal of Software and Systems Modeling*, May 2006.

[15] A. K. Nørgaard, L. Pedersen, and P. Strøiman. Method for generating a workflow on a computer, and a computer system adapted for performing the method. Patent, 05 2005. US 6895573.

[16] C. A. Olarte and F. D. Valencia. Universal concurrent constraint programming: Symbolic semantics and applications to security. In *23rd Annual ACM Symposium on Applied Computing (SAC)*, 2008.

[17] M. Pesic and W. van der Aalst. A Declarative Approach for Flexible Business Processes Management. *Lecture Notes in Computer Science*, 4103:169, 2006.

[18] F. Puhlmann and M. Weske. Using the Pi-Calculus for Formalizing Workflow Patterns. *BPM*, 3649:153–168, 2005.

[19] V. Saraswat. *Concurrent Constraint Programming*. MIT Press, 1993.

[20] W. van der Aalst and M. Pesic. DecSerFlow: Towards a Truly Declarative Service Flow Language. *Lecture Notes in Computer Science*, 4184:1, 2006.

[21] H. Vieira, L. Caires, and J. Seco. The Conversation Calculus: A Model of Service-Oriented Computation. In *Programming languages and systems: 17th European Symposium on Programming (ESOP)*, page 269, Budapest, Hungary, 2008. Springer-Verlag New York.