Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 162 (2006) 233–236
www.elsevier.com/locate/entcs

Process Calculi and Peer-to-peer Web Data Integration

Sergio Maffeis

Department of Computing, Imperial College London.
maffeis@doc.ic.ac.uk.

Abstract

Peer-to-peer systems exchanging dynamic documents through web services are a simple and effective platform for data integration on the Web. Dynamic documents can contain both data and declarative references to external sources, in the form of links, service calls, or coordination scripts. XML standards and industrial platforms for web services provide a wide technological basis for building such systems. We argue that process algebras are a promising tool for studying and understanding their formal properties.

Keywords: Process calculi, XML, peer-to-peer, data integration, Web services, orchestration.

1 Dynamic Web Data

The Internet is a global network used in daily activities to find information, communicate ideas, conduct business and carry on distributed computations. Due to the very large size of the Web, in order to fully exploit its potential there is a need for scalable mechanisms for organizing and manipulating all the available information. Peer-to-peer architectures help us face the issue of scalability, and technologies such as XML and Web services facilitate the development of distributed applications. In brief, XML is a standardized data model and format targeted at inter-operability, and Web services are sites which are designed to be used by applications rather than humans. XML is used both for the representation of data and for invoking, describing and discovering Web services (SOAP, WSDL and UDDI).

Web-based systems for data integration constitute a challenging application for these technologies, not only due to the vast heterogeneity of Web data sources, but also because of the possibly complex communication patterns which arise in translating the declarative high-level operations of these systems (mostly data queries) into low-level execution plans, which may involve the recursive invocation of queries at different sites.

doi:10.1016/j.entcs.2005.12.087

Let us consider now the generic structure of these systems. A variable number of interconnected peers, all sharing a similar internal structure, and each one identified by a unique name, compose a network. Peers can communicate with each other using a common protocol and, due to the level of abstraction, connectivity is not restricted by administrative domains or firewalls. Networks are *open* in the sense that new peers can always join in, and that external hosts can interact, in a restricted way, with the existing peers. Each peer acts both as a provider and a consumer of information. It contains a data repository, an internal working space where agents carry out local computations and a network interface providing remote communication and services to other peers. Agents can communicate with each other, query and update the local repository and, when the architecture is predisposed, migrate to other peers to continue execution. The repositories export a view of their data in a semi-structured format, which contains enough meta-information to facilitate queries also when data does not obey a fixed schema. Data can contain scripted agents, queries, and declarative references to services provided by other peers. We refer to this scenario as *dynamic Web data*.

As a first example, we can consider hosts connected to the Internet running both a server and a browser. Hosts use the HTTP protocol to interact with each other, either requesting or providing information. Hyperlinks and client-side scripts embedded in HTML pages are examples of how data can be dynamic. Let us consider now two concrete examples: Active XML [3,5] and ObjectGlobe [7]. In the Active XML system, data in the repository can contain service calls (requests to remote peers) with parameters which are either explicit or expressed in terms of path expressions (queries on the local data). Services can run arbitrary code, but typically consist of an OQL query-update expression on the local repository. One interesting source of flexibility in Active XML is the choice of when to invoke the service calls stored in a piece of data. They can be invoked either periodically, or when data is fetched by the server, or when it is returned to the client. In the ObjectGlobe system instead, the emphasis is on executing complex queries on distributed data sources by discovering what peers provide operators or data relevant to the task at hand, and then dispatching the corresponding sub-queries to the relevant sites. In this case dynamic data can contain queries to other repositories, and services are geared towards coordination.

Despite the practical usefulness of these system, a specialist in process calculi is likely to be struck by the potential of dynamic Web data which still lies unexpressed. For example, there are often drastic restrictions on the amount of dynamic behaviour that data can exhibit, and the range of data flows that can be specified is quite limited. Typical problems that may have determined these restrictions are that it is hard to manage persistent state across different service call invocations, and to regulate the interaction between concurrent processes. Moreover, security and efficiency are of primary interest, hence sometimes expressivity has been sacrificed in favour of simplicity. Process algebras have proven to be a convenient setting for studying similar problems, and we expect a conspicuous benefit from applying their techniques to architectures for dynamic Web data.

A Process Algebraic Approach. Process calculi provide a simple and expressive framework for reasoning about the properties of concurrent and distributed systems. The π-calculus of Milner, Parrow and Walker [19] is a terse language for describing the behaviour of concurrent systems, endowed with a rich body of theoretical results. It is the basis for many other calculi that target specific aspects of concurrent and distributed systems. Just to mention some, the spi-calculus [2] and theapplied π-calculus [1] have been used to study security protocols, the distributed π-calculus [15] for controlling the access to resources, the Ambient Calculus [10] to study mobile computations across administrative domains and the Join-calculus [12] has been used as a basis for distributed implementations.

Sahuguet *et al.* [21], in some preliminary work eventually leading to the design of the ubQL query language [20], first applied ideas from the π-calculus to distributed query systems, which can be considered as precursors of these data integration platforms. In joint work with Gardner [14], we have defined the X$d\pi$-calculus explicitly to reason about dynamic Web data. X$d\pi$ terms represents networks of peers where each peer consists of an XML data repository and a working space where processes are allowed to run. Our processes can be regarded as agents with a simple set of functionalities; they communicate with each other, query and update the local repository, and migrate to other peers to continue execution. Process definitions can be included in documents, and can be executed by other processes. Adapting techniques from the asynchronous π-calculus [16] and the higher-order π-calculus [22,17], we have studied behavioural equivalences for X$d\pi$.

Process calculi have been involved in other research relevant to dynamic Web data, more focused either on Web service orchestration or on XML manipulation. For example, Bruni *et al.* [9] formalized, and translated into the Join-calculus, an operational model of distributed transactions inspired by Microsoft BizTalk; Laneve and Zavattaro [18] studied an extension of the asynchronous π-calculus with loosely coupled transactions called webπ; and Ferrara [11] gave a bidirectional translation between the BPEL4WS orchestration language for web services and the LOTOS process algebra (in fact, the π-calculus has influenced the design of XLANG, a precursor of BPEL4WS). On the XML front, Bierman and Sewell [6] defined a strongly typed XML scripting language (called Iota) with concurrency primitives inspired by the π-calculus, and showed that it can be used to program Home Area Networks. Castagna *et al.* [13] applied the semantic subtyping approach of $\mathbb{C}$Duce to $\mathbb{C}\pi$, a π-calculus extended with pattern matching and tuple values. Using an encoding, they represented XML values in $\mathbb{C}\pi$, achieving a degree of expressivity similar to that of $\mathbb{C}$Duce pattern matching. Brown *et al.* [8] defined an extension of the π-calculus with native XML datatypes called πDuce, and considered a higher order extension which enables dynamic content in documents. Finally, Acciai and Boreale [4] proposed an extension of the asynchronous π-calculus with code mobility and ML-like pattern matching, using a type system to ensure basic safety properties.

We hope that this introduction may serve as a source of inspiration for further research on peer-to-peer Web data integration, a setting where many techniques, which studied in isolation may have a prominently academic appeal, can be com-

bined together to improve the design of commercial applications.

References

[1] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. *ACM SIGPLAN Notices*, 36(3):104–115, March 2001.

[2] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation* 148(1999):1-70.

[3] Abiteboul, S. et al. Active XML primer. INRIA Futurs, GEMO Report N. 275, 2003.

[4] L. Acciai and M. Boreale. XPi: A typed process calculus for XML messaging. In *Proceedings of FMOODS'05*, 2005.

[5] Benjelloun, O. Active XML: A data-centric perspective on Web services. PhD thesis, Universtity of Paris XI, 2002.

[6] G. Bierman and P. Sewell. Iota: a concurrent XML scripting language with application to Home Area Networks. University of Cambridge Technical Report 557, jan 2003.

[7] Braumandl, R. et al. ObjectGlobe: Ubiquitous query processing on the internet. *VLDB Journal: Special Issue on E-Services*, 2002.

[8] A. Brown, C. Laneve, and G. Meredith. PiDuce: a process calculus with native XML datatypes. Unpublished manuscript, 2004.

[9] R. Bruni, C. Laneve, and U. Montanari. Orchestrating transactions in join calculus. In *Proceedings of CONCUR '02*, LNCS 2002.

[10] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.

[11] Andrea Ferrara. Web services: a process algebra approach. In *Proceedings of ICSOC '04*, ACM Press 2004.

[12] C. Fournet and G. Gonthier. The reflexive CHAM and the join-calculus. In *Proceedings of POPL'96*, ACM Press 1996.

[13] R. De Nicola G. Castagna and D. Varacca. Semantic subtyping for the pi-calculus. Proceedings of LICS'05, IEEE, 2005.

[14] P. Gardner and S. Maffeis. Modelling dynamic Web data. *Theoretical Computer Science*. 342:104–131, 2005.

[15] M. Hennessy and J. Riely. Resource access control in systems of mobile agents. *Information and Computation*, 173:82–120, 2002.

[16] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 151(2):437–486, 1995.

[17] A. Jeffrey and J. Rathke. Contextual equivalence for higher-order pi-calculus revisited. Computer Science Report 04/2002, University of Sussex, 2002.

[18] C. Laneve and G. Zavattaro. Foundations of web transactions. In *Proceedings of FoSSaCS'05*, LNCS 3441, 2005.

[19] R. Milner, J. Parrow, and J. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–40,41–77, 1992.

[20] Sahuguet, A. ubQL: A Distributed Query Language to Program Distributed Query Systems. PhD thesis, University of Pennsylvania, 2002.

[21] A. Sahuguet, B. Pierce, and V. Tannen. Distributed Query Optimization: Can Mobile Agents Help? Unpublished draft.

[22] D. Sangiorgi. Expressing mobility in process algebras: First-order and higher-order paradigms. PhD thesis, University of Edinburgh, 1992.