Available online at www.sciencedirect.com
SCIENCE DIRECT
ELSEVIER
Electronic Notes in Theoretical Computer Science 105 (2004) 5–9
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Web Services and Models of Computation*

Ugo Montanari[a]

[a] *Dipartimento di Informatica, Università di Pisa*
via F. Buonarroti 2, 56127 Pisa, Italy
{ugo}@di.unipi.it

Abstract

We review the challenges of providing Web Services, especially those this new technology will impose on models of computation.

Keywords: Web services, XML, software architecture, Fusion calculus, π-calculus, Petri nets, mobile synchronization.

Web Services

While the web is presently mainly used for retrieving remote informations and documents, in the near future a variety of distributed applications will offer, discover and negotiate suitable web services via standard XML interfaces. This scenario is extremely challenging from both the practical and the theoretical viewpoint. In fact, it requires on the one side to redesign in most cases the software architectures of the relevant applications, and on the other it makes the existing foundations of computer science often inadequate in terms of expressiveness, efficient implementation, verification and security. In this talk we will focus on some of the new requirements web services impose on models of computation, and on some existing proposals for their fulfillment.

* Research supported in part by the EU-FET project IST 2001-33100 PROFUNDIS.

doi:10.1016/j.entcs.2004.05.009

Some Requirements for Models of Computations

Mobility and Fusions

A first requirement concerns the ability to express locations, and mobility among them, in a general open ended framework. Calculi like π-calculus, JOIN, Ambients have been proposed in the last several years to fulfill these needs, and they have yielded the foundations for experimental programming languages like BizTalk [16], HighWire [15] and Polyphonic C$^\sharp$ [1]. We find particularly interesting Fusion calculus, in both the original version by Parrow and Victor [17] and in the version with explicit fusions by Gardner, Laneve, and Wischik [12]. Fusions are appealing in several variants of name mobility: e.g. the *open* construct of Ambients can be conveniently modeled as the fusion of two locations. When explicit, fusions behave like messages, and their effect is propagated asynchronously to the rest of the world.

A recent paper [4], analyzes the expressiveness of Fusion calculus when compared with (open) π-calculus, and concludes that neither of them is more expressive than the other. The reason is that while the former has powerful fusion constructs which are missing in the π-calculus, the restriction operator of the latter cannot be simulated by scope restriction of the Fusion calculus. In fact, names extruded by Fusion calculus can still be fused with existing names, while π-calculus fresh names are guaranteed to be fresh forever. This possibility is considered as a significant lack of expressiveness for those applications, like security protocols, where fresh names are used for modeling private channels, session keys and nonces. A new calculus with two binders, called DFusion, is then proposed, which is an extension of both open π-calculus and Fusion calculus.

Committed Negotiations

Most models of computation proposed for global computing are fully asynchronous, i.e. communication is via message sending/receiving and variables are shared only locally. This level of abstraction is adequate for middleware and network aware programming, but a higher level might be more convenient for most web service applications. In particular, negotiations should be committed or compensated at a global, system level, without leaving the burden of synchronization on the participants. This is even more mandatory if one considers cases where new, unregistered participants can enter and leave the negotiation at any time.

A foundational model for a general notion of committed computations is a version of Petri nets called *zero-safe* nets [7]. A zero safe net is equipped

with both a Petri-like operational semantics and with an abstract behavior represented by a place-transition net where transactions of the zero-safe net become ordinary transitions. The committed behavior can be defined using an extended two-phase commit protocol, which can be easily specified and implemented in the JOIN calculus [5]. Moreover, taking advantage of the similarity between Petri nets and the JOIN calculus, the zero safe construction and the extended two-phase commit protocol can be lifted to the JOIN calculus itself, obtaining a new calculus equipped with committed negotiations and compensations called *committed* JOIN or cJOIN [6]. A similar construction applies to LINDA [8]. An approach for modeling long-running transactions in the π-calculus is presented in [3].

Service Invocation as Constraint Solving

When web services are published, suitable constraints must be defined for their use, and then checked when services are invoked. In some cases, more than two sites may be involved in establishing a connection, e.g. when routing or access paths are involved. In the dynamic, open situation typical of web services, a site cannot anticipate the requests of the other sites, let alone solve the global constraint that arises in this way. Thus suitable network services must be provided to do this job. Another aspect of the same problem is that there might be non-crisp situations where a constraint is not considered as solved in a yes or no way, but solved with some cost, or some probability, or according to several criteria at the same time.

A convenient representation of scenarios where several processes should synchronize on various channels at the same time can be obtained using *Synchronized Hyperedge Replacement* [13,14,10]. This approach allows to easily represent both multiple synchronizations, as required by Ambients [10] and mobility as typical of π-calculus [14]. Mobility within multiple synchronization yields name fusion problems solved via unification. Constraint programming in the yes/no fashion can be extended to non-crisp cases where the evalutation of a constraint yields a value on a *c-semiring*, i.e. a semiring with certain additional axioms [2]. The multiplicative and additive operations on the semiring correspond to imposing two constraints on the same problem and of choosing the "best" between two constraints respectively. Synchronized Hyperedge Replacement and c-semiring constraint solving can be combined to obtain non-crisp constraint solving in multiple syncronizations [11] (unification is a special case), for instance for computing the most convenient path with the necessary access rights [9].

Acknowledgments

The work outlined here is carried on in collaboration with Stefano Bistarelli, Michele Boreale, Roberto Bruni, Marzia Buscemi, Rocco De Nicola, Gianluigi Ferrari, Dan Hirsch, Paola Inverardi, Cosimo Laneve, Hernan Melgratti, Rosario Pugliese, Francesca Rossi and Emilio Tuosto.

References

[1] N. Benton, L. Cardelli, and C. Fournet. Modern concurrency abstractions for C♯. In B. Magnusson, editor, *Proceedings of ECOOP 2002*, volume 2374 of *Lect. Notes in Comput. Sci.*, pages 415–440, Spain, June 2002. Springer Verlag.

[2] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2):201–236, March 1997.

[3] L. Bocchi, C. Laneve, and G. Zavattaro. A calculus for long-running transactions. In *Proceedings of FMOODS'03*, Lect. Notes in Comput. Sci. Springer Verlag, 2003. To appear.

[4] M. Boreale, M. Buscemi, and U. Montanari. The distinctive fusion calculus. Unpublished draft, 2003.

[5] R. Bruni, C. Laneve, and U. Montanari. Orchestrating transactions in join calculus. In L. Brim, P. Jancar, M. Kretinsky, and A. Kucera, editors, *Proceedings of CONCUR 2002*, volume 2421 of *Lect. Notes in Comput. Sci.*, pages 321–336. Springer Verlag, 2002.

[6] R. Bruni, H. Melgratti, and U. Montanari. Flat Committed Join in Join. In F. Honsell, M. Lenisa and M. Miculan, editors, *Proceedings of COMETA 2003: Computational Metamodels*, ENTCS, to appear.

[7] R. Bruni and U. Montanari. Zero-safe nets: Comparing the collective and individual token approaches. *Inform. and Comput.*, 156(1-2):46–89, 2000.

[8] R. Bruni and U. Montanari. Concurrent Models for Linda with Transactions. *Math. Struct. in Comp. Sc.*, to appear.

[9] R. De Nicola, G. Ferrari, U. Montanari, R. Pugliese, and E. Tuosto. A formal basis for reasoning on programmable QoS. In: Nachum Dershowitz, Ed., Verification–Theory and Practice: Proceedings of an International Symposium in Honor of Zohar Manna's 64th Birthday, Springer LNCS, to appear.

[10] G. Ferrari, U. Montanari, and E. Tuosto. A lts semantics of ambients via graph synchronization with mobility. In *7th Italian Conference on Theoretical Computer Science – ICTCS'01*, volume 2202 of *Lecture Notes in Computer Science*. Springer, 2001.

[11] G. Ferrari, U. Montanari, and E. Tuosto. Graph-based Models of Internetworking Systems. In: Bernhard K. Aichernig and Tom Maibaum, Eds., UNU/IIST 10th Anniversary Colloquium, Springer LNCS, Vol. 2757 pp.242-266.

[12] P. Gardner, C. Laneve, and L. Wischik. The fusion machine (extended abstract). In *Proc. of CONCUR '02*, LNCS 2421. Springer-Verlag, 2002.

[13] D. Hirsch, P. Inverardi, and U. Montanari. Reconfiguration of software architecture styles with name mobility. In A. Porto and G.-C. Roman, editors, *Coordination 2000*, volume 1906 of *LNCS*, pages 148–163. Springer Verlag, 2000.

[14] D. Hirsch and U. Montanari. Synchronized hyperedge replacement with name mobility: A graphical calculus for name mobility. In *12th International Conference in Concurrency Theory (CONCUR 2001)*, volume 2154 of *LNCS*, pages 121–136, Aalborg, Denmark, 2001. Springer Verlag.

[15] L. G. Meredith, S. Bjorg, and D. Richter. Highwire Language Specification Version 1.0. Unpublished manuscript.

[16] Microsoft Corp. Biztalk Server - `http://www.microsoft.com/biztalk`.

[17] J. Parrow and B. Victor. The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes. In *Proc. of LICS'98*. IEEE Computer Society Press, 1998.