A graph-based framework for Fusion Calculus

Ivan Lanese

Dipartimento di Informatica, Università di Pisa
e-mail:`lanese@di.unipi.it`

Our work is in the field of *formal models* for *concurrent* and *distributed* systems.
A system is said to be distributed if it has multiple components deployed in different (physical or logical) places (a local area network for instance). A system is said to be concurrent if it has multiple threads of execution computing at the same time (a multiprocessor for instance).
Since interactions between different components of a system can be very complex and the system itself can be huge, formal methods are necessary in order to drive the design and the implementation of the system and to allow to formally prove properties of the system (this is necessary for safe-critical systems, such has a nuclear power plant control system).
A classical approach to this kind of problems is the *process algebras* one, where a system is represented by a term in a term algebra that contains suitable operators such as parallel composition and elementary processes as constants. This approach has the advantage of having a compositional semantics (the semantics of the system can be derived from the semantics of components) and of having a lot of developed standard techniques to deal with common issues. We mention SOS semantics, which allows a formal definition of the behaviour of open systems (systems which interact with the environment), and bisimulation, which is a technique that allows to identify systems which are equal with respect to the allowed interactions with the environment.
We are interested in particular in systems with *mobility*, that is systems whose physical components change their locations during the computation. We can have either physical mobility (for instance in the case of mobile phones or laptops) or code mobility (when only the code is moved, as in Java applets that can be downloaded from a remote server and executed locally).
The corner stone in the field of process algebras with mobility is the π-*calculus* [3], which combines a simple mathematical structure and a great expressivity, which allows one to model very different kinds of systems. Thanks to these properties the π-calculus has been used in a lot of contexts (security, program verification, ...) and in its typed version is the base of some real programming languages (such as BizTalk).
In this work we will study the *Fusion Calculus* [4], which can be seen as the successor of the π-calculus, since it contains the π-calculus as a subcalculus. The important aspect of this extension is that it simplifies and makes more symmetric the π-calculus while making

it more powerful and more apt to model object-oriented and shared memory systems.
We will analyze the relations between Fusion Calculus and *Synchronized Hyperedge Replacement* (SHR) [1, 2], which is a model of distributed systems based on hypergraphs and *hypergraph transformations*. Here computational entities (such as workstations in a network) are represented by hyperedges and links among them by shared nodes. Transformations between two states of the system are obtained by specifying how single hyperedges evolve and by coordinating the single rewritings by imposing constraints on shared nodes. The possible evolutions of the system are determined by solving the distributed constraints problem. Furthermore references to nodes can be transmitted from one hyperedge to another during computation, what allows to model mobility.
This model has the advantage of having an intuitive graphical representation (while process algebras are usually difficult to understand for non specialized people) and at the same time a formal semantics based on syntactic judgements.
We will show that the fundamental computational mechanism in both cases is unification and we will prove a bijective correspondence between a restriction of the Fusion Calculus (which contains however the most interesting part of it) and a specialized version of SHR. This specialized version is required since standard SHR is inherently concurrent whereas the semantics of Fusion Calculus is interleaving (at each step at most one action is done). The translation is very natural and compositional. A result of this work is that we can derive for free a concurrent semantics for Fusion Calculus, which can be obtained by applying the same translation and then executing the hypergraph transformations using the standard rules.
A possible extension of this work is using the translation in order to transfer the work done on bisimulation properties of the Fusion Calculus in the field of SHR, where the approach is not quite developed. Furthermore it is possible to find a translation to an SHR sytem which uses a different kind of synchronization, which is strongly related to logic programming, what can be used for implementation purposes.

References

[1] Pierpaolo Degano and Ugo Montanari. A model for distributed systems based on graph rewriting. *Journal of the ACM (JACM)*, 34(2):411–449, 1987.

[2] GianLuigi Ferrari, Ugo Montanari, and Emilio Tuosto. A lts semantics of ambients via graph synchronization with mobility. In Simona Ronchi Della Rocca Antonio Restivo and Luca Roversi, editors, *Proc. of ICTCS'01*, volume 2202 of *LNCS*, pages 1–16. Springer, October 2001.

[3] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Journal of Information and Computation*, 100:1–77, September 1992.

[4] Björn Victor. *The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes*. PhD thesis, Dept. of Computer Systems, Uppsala University, Sweden, June 1998.