Graphs, tiles and the ambient calculus*

Fabio Gadducci and Ugo Montanari

Dipartimento di Informatica, Università di Pisa, via Buonarroti 2, I-56127 Pisa, Italia.
`{gadducci,ugo}@di.unipi.it`

Latest years witnessed a renewed interest in the use of graphical formalisms as a suitable implementation technique, perhaps after the advances on so-called optimal implementation of λ-calculus [1]. Among other proposals, many authors discussed graphical presentations for calculi with name mobility, in particular for the π-calculus [9]. These descriptions usually introduce a syntactical notation for graphs, then mapping processes into graphs *via* that notation. With a few exceptions [10], the resulting graphical structures are eminently hierarchical (that is, roughly, each node/edge/label is itself a structured entity, and possibly a graph), thus forcing the development of ad-hoc mechanisms for graph rewriting, in order to simulate process reduction.

In [6] we presented a general proposal for mapping processes of calculi with name mobility into unstructured, non-hierarchical graphs. Mobile ambients [3] were chosen as the leading example, partly for its rising popularity in the community, while still lacking an analysis of its concurrency features; and partly because the complex name handling presented by its reduction rules highlights the power of the graphical framework. In our opinion, the intuitive appeal of non-hierarchical graphs, and the local nature of the associated rewriting mechanism, may help casting some light on the distributed features of the calculus. So, after proving that our encoding of processes into graphs is sound and complete (i.e., that two processes are structurally equivalent if and only if the corresponding graphs are isomorphic), and that it is faithful with respect to the reduction semantics (i.e., that standard graph rewriting techniques may be used to simulate reduction steps on processes by sequences of rewrites on their encodings), we used the well-established concurrent semantics for graph rewriting in order to discuss about concurrent reductions as well as *interference* in mobile ambients.

Roughly, the graphical encoding for calculi with name mobility is obtained via the construction of the (hyper-)tree associated to a process, whilst parallel composition is simulated by tree (disjoint) union, plus a suitable mechanism for coalescing nodes representing the same variable, or denoting the top operators of each process. Additionally, two auxiliary operators, `go` and `idle`, are introduced as a syntactical device for detecting the status of a node, i.e, for checking if a given subgraph (representing a reducible process) is allowed to fire. It is relevant that that same encoding, up to the auxiliary operators, could be used to provide a *tile semantics* for the mobile ambient calculus.

* This work has been partially supported by the Italian MIUR project COMETA (*Computational Metamodels*); and by EU within the FET – Global Computing initiative, project AGILE IST-2001-32747 (*Architectures for mobility*). The funding bodies are not responsible for any use that might be made of the results presented here.

Tile logic [5] has been proposed as a general framework for the specification of rule-based systems, whose actual behaviour relies on the notion of *synchronisation* and *side-effect*. The main idea is to enrich each rewriting rule with an observation, carrying information about the possible behaviour of its subcomponents, that is, imposing a dynamic constraints to to the terms to which it can be applied. Th resulting formalism is thus well-suited for representing generic (open) configurations of reactive systems. As a natural extension of rewriting logic [8], it has been proved adequate as a semantical framework for encoding various other formalisms, including calculi with localities [4] or those used for system specifications where *name handling* is a key issue [2].

A main strength of the framework with respect to similar proposals (as noted in [7]), is that the composition mechanism for tiles is actually parametric with respect to the chosen formalism for representing configurations. Hence, the graphical encoding previously developed may then be used again as a basis for mapping mobile ambients into tiles. What is actually needed is a further simplification, obtained by dropping the auxiliary operators, since the coordination features of tiles allow for recasting the reduction semantics. Such a simplification is precisely one of the distinguishing characteristic of tile logic, and one of the main reasons for its introduction. The talk illustrates the graphical encoding and how the related tile system is built, and hints at how the construction may provide a different encoding, where the auxiliary operators are removed.

References

1. A. Asperti and S. Guerrini. *The Optimal Implementation of Functional Programming Languages*. Cambridge University Press, 1998.
2. R. Bruni and U. Montanari. Dynamic connectors for concurrency. *Theoret. Comput. Sci.*, 281:131–176, 2002.
3. L. Cardelli and A. Gordon. Mobile ambients. *Theoret. Comput. Sci.*, 240:177–213, 2000.
4. G. Ferrari and U. Montanari. Tile formats for located and mobile systems. *Information and Computation*, 156:173–235, 2000.
5. F. Gadducci and U. Montanari. The tile model. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 133–166. MIT Press, 2000.
6. F. Gadducci and U. Montanari. A concurrent graph semantics for mobile ambients. In S. Brookes and M. Mislove, editors, *Mathematical Foundations of Programming Semantics*, volume 45 of *Electronic Notes in Computer Science*. Elsevier Science, 2001.
7. F. Gadducci and U. Montanari. Comparing logics for rewriting: Rewriting logic, action calculi and tile logic. *Theoret. Comput. Sci.*, 285:319–358, 2002.
8. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoret. Comput. Sci.*, 96:73–155, 1992.
9. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes. Part I and II. *Information and Computation*, 100:1–77, 1992.
10. N. Yoshida. Graph notation for concurrent combinators. In T. Ito and A. Yonezawa, editors, *Theory and Practice of Parallel Programming*, volume 907 of *Lect. Notes in Comp. Science*, pages 393–412. Springer, 1994.