Models of Computation: A Tribute to Ugo Montanari's Vision

Roberto Bruni[1] and Vladimiro Sassone[2]

[1] Dipartimento di Informatica, Università di Pisa, Italia
[2] ECS, University of Southampton, UK

Ugo Montanari's Models of Computation

Ugo's research activity in the area of Models of Computation (MoC, for short) has been prominent, influential and broadly scoped. Ugo's trademark is that undefinable ability to understand and distill computational aspects into new models as if you were reading them out of some evident connection between well-know models: only, most often, that connection is really visible only after Ugo shows the way. Like experienced sailors have trusted compasses and sextants to help them find the best routes to harbour, Ugo relies on a bag of favourite tools which he has used along the years to deliver a variety of contributions to the MoC area. To mention just three (in alphabetic order): algebraic techniques, concurrency theory, and unification mechanisms.

In this introductory contribution we would like to recall some of the influential MoC models put forward by Ugo which cut across the three approaches. Before doing that, it is worth devoting some space to discuss the three aspects separately. Notably, the use of category theory is a pervasive common trait.

Algebraic techniques. By algebraic techniques we refer broadly to the use of universal algebras and initial model semantics; of universal coalgebras and final semantics; and of bialgebras. Many interesting papers witness Ugo's leading role in exploiting algebraic techniques during his entire scientific career. Indeed, his contributions are too many to mention all in the space allocated to this overview; we shall therefore attempt to convey the sense of Ugo's broad-spectrum contribution by recapping only a few key results.

Reference [43] is the first paper on final, observational semantics in abstract data types, and the main reference for one of the MoC contributed papers in this volume. It presented several key insights in software specification and development for the first time, like the separation between given sorts and newly specified ones, whereby the given sorts lay the ground to define the observable behaviour for the new sorts. Another key suggestion is that the specification of new data types is often partial —in the sense that it may include "*don't care*" cases— and that many realisations can exist that exhibit equivalent observable behaviour but are not isomorphic. In fact, [43] shows that the isomorphism classes of observably equivalent algebras conforming to the partial specification form a complete lattice, yielding a so-called loose semantics.

Possibly the best known of Ugo's papers, [52] exposes the underlying monoidal structure of the category of Petri net computations. The title itself is revealing: *Petri nets are monoids*. Besides doing what it says on the tin, this paper opened a long-lasting and fruitful collaboration with José Meseguer, and a research line on the initial semantics of

P. Degano et al. (Eds.): Montanari Festschrift, LNCS 5065, pp. 503–509, 2008.

computational models in which, after [33], we got deeply involved ourselves [55,56,17,24,18]. The key insight is that by lifting the algebraic structure of (machine) states to the level of computations via a so-called free construction, one can gain a deeper understanding of the axioms which regulate equivalent computations (or processes), an idea that is also the basis of Meseguer's Rewriting Logic [51]. This theme of lifting the algebraic structure of states to the level of concurrent computations has motivated the study of Structured Transition Systems [30,39,37], while finding axiomatisations of computational structures has been reconsidered in [49,16,27,9,11].

A more recent result in coalgebraic semantics [57] has paved the way to the efficient verification techniques for the π-calculus [35], and to the bialgebraic semantics of fusion calculus [25]. The technique proposed in [57] addresses the issue of finding a suitable setting to develop a coalgebraic semantics for the π-calculus, so as to characterise minimal process realisations. The key difficulty is the proper handling of fresh names, tackled by exploiting a category of name-permutation algebras which underpins the coalgebraic treatment of the π-calculus operational semantics.

Concurrency theory. Concurrency theory encompasses many different techniques and approaches, ranging from bisimilarity and contextual equivalences to event structure semantics. It is harder here to make a representative selection of a few seminal papers, because of the quality and volume of Ugo's work in the area of concurrency models.

Given our previous lives in 'Petri-land,' we cannot help but mention the work on unfolding semantics that generalised Winskel's approach from the class of safe nets to a wide class of place/transition nets [54]. An unfolding semantics accounts for a full fledged view of the admissible computations, including concurrency, causality and conflict aspects: the so-called "truly concurrent" semantics. Exploiting mathematical tools from category theory, the main result establishes that a chain of adjunctions (a suitable categorical notion indicating that the corresponding construction is as good as possible) leads from the category of Petri nets to the category of prime event structures, which is equivalent to the category of coherent finitary prime algebraic domains (because of this, the unfolding approach is sometimes referred to as a denotational semantics).

More recently, it was shown that such event structure semantics can be extended to a more sophisticated setting of contextual nets and graph transformation systems, where e.g., multiple concurrent read accesses to the same resource and inhibiting conditions for the occurrence of certain events can be accounted for. The price to pay was the introduction of more complex event structures [3,28,2,4,5]. Significantly related is also [1]. The extension has made it possible to provide event structure semantics to mobile calculi for free by encoding them in graph transformation systems [14,15].

The paper [21] presents a mathematical setting building upon some analogies in the representation of names, locations and causal links as shared entities. Such a uniform treatment of different concepts opens the way to the definition of a general-purpose meta-model to be instantiated to several cases of interest. The main result shows that the framework can be applied to the basic parallel processes with weak synchronisation, by defining an operational semantics that accounts for concurrency aspects and a causal abstract semantics and showing it equivalent with bisimilarity via "causal trees" [32].

It is finally worth to mention Ugo's work on transactional extensions of concurrent frameworks [44,20,12,13,7].

Unification mechanisms. Logic programming and its extensions, in particular with concurrent constraints, are one of the long-term research interests of Ugo's. In logic programming, resolution steps are based on the notion of unification between the head of a logic clause and a selected atomic sub-goal. The Martelli-Montanari algorithm [50] is arguably the best known unification algorithm for constructing the "most general unifier" (mgu) between a sub-goal and the head of a clause. Since those brilliant beginnings, the view of unification as an elegant coordination mechanism has been a recurrent source of inspiration in Ugo's work on MoC. We mention here three cases.

Reference [30] builds on the view of mgu as a categorical "equaliser:" clauses are seen as rewrite rules whose variables can be further instantiated freely, and the computational model of a logic program is a suitable 2-category. The interesting point is that the 2-cells of the 2-category are equipped with an algebraic structure that captures some concurrency aspects. If there exists a refutation for the goal G with computed answer substitution θ, then in the 2-category model we can find a refutation for the goal $G\theta$ but not necessarily one for G. This situation is improved in [22], where the mgu is expressed as a categorical "pullback" square and double-categories are considered instead of 2-categories. This setting can account for the dynamic creation of fresh variables and deal with the computed answer substitutions instead of just the correct answer substitutions.

The ideas in [22] are further developed in [6], where logic programming "resolution rule" is generalised to MoC tailored to the needs of the general server-to-client bindings required by the service oriented applications. When a new service is discovered, not only it must adapt to the client, e.g., accepting a list of parameters, but vice versa the client too must sometimes adapt to the server in order to establish the connection. Then, the mgu represents the minimal possible adaptation that should be sought in order to minimise the possible degradation.

Combined approaches. Much of Ugo's scientific thinking can be characterised as the aspiration to combine modelling elements so that the combination of the parts is more expressive and flexible than their mere sum. Below we point out some examples.

The CHARM [31], Concurrency and Hiding in an Abstract Rewriting Machine, is an abstract machine that combines algebraic techniques typical of process calculi with the experience in constraint logic programming and graph transformation systems. Characteristic of the CHARM is the ability to capture the essence of concurrent computations in systems composed by a global, shared part and locally distributed resources.

GDS [26,34], Grammars for Distributed Systems, combines distributed computation based on Hoare synchronisation with concurrent histories. This model later evolved in Synchronized Hyperedge Replacement, SHR [45,46,40,36,48], where different synchronisation mechanisms are considered together with node merging and splitting.

HD-automata [57] (see also the section on Software Verification in the present volume), for History Dependent Automata, are an extension of ordinary automata aimed to endow them with name handling features: states and transition labels may contain names which can represent, e.g., communication channels or locations in distributed systems. Each transition establishes a correspondence between the names in the source state, those in the label and those in the target state. HD-automata permit an adequate representation of the behavior of calculi with name mobility, as names can be garbage-collected and reused to identify 'verification-friendly' processes semantics.

The Tile Model [41,53,58,19,29,38,8,47,16,42,10,23] combines the modularity of Structured Transition Systems with Meseguer's Rewriting Logic approach. While rewrite rules in Rewriting Logic can be applied in any context and with any actual parameters, the Tile Model allows rewritings to be inhibited under certain contexts. In category theory, this correspond to move from 2-categories to double-categories. Moreover, as tiles have been designed around concurrent systems, it is common to consider a monoidal structure of states that gives raise to a monoidal double-category of computations. Thanks to these features, the Tile Model offers a framework where the specification of process calculi with name passing, causality and locality becomes uniform and several important results can be accounted for at the meta-theoretical level.

Papers on Models of Computation in This Volume

The six contributed papers in this section of the present volume cover several of Ugo's favourite topics; other papers on models of computation are included in other chapters dealing with more specific contexts and applications.

Martín Abadi: Automatic mutual exclusion and atomicity checks. This contribution presents a calculus for studying the Automatic Mutual Exclusion (AME) programming model. Roughly, the AME calculus consists of a concurrent lambda calculus with references, extended with constructs for thread spawning, yielding, blocking and atomic execution. A type system ensures that atomic blocks are not violated through yield executions. The main results show soundness and progress theorems.

Samson Abramsky: Petri nets, discrete physics, and distributed quantum computation. This inspired paper builds interesting connections between separate fields, and does so by building upon some of Ugo's best known work. In fact, it describes analogies between Petri Nets, monoidal categories with additional structure, and quantum mechanics (in particular quantum information).

Filippo Bonchi, Maria Grazia Buscemi, Vincenzo Ciancia and Fabio Gadducci: A category of explicit fusions. The paper introduces a suitable category E of equivalence relations and shows it suitable to represent (abstract) syntax and semantics (via an endofunctor B on Set^E) of the calculus of explicit fusions. The main result gives a bijection between inside-outside bisimulations and coalgebraic bisimulations for B.

José Luiz Fiadeiro: What do semantics matter when the meat is overcooked? This paper presents a model for configuration management of service-oriented applications modelled with the language developed by the EU funded Sensoria project. The model makes use of various of Ugo's favourite ingredients: roughly, business configurations are represented as graphs; constraint systems play the role of business policies; a module requiring a set of services is seen as a clause in logic programming style; the reconfiguration that happens when a service is called for instantiation (via the usual service-oriented mechanism of discovery, selection, and binding) is modelled by a sort of resolution.

Nicoletta Sabadini and Robert Walters: Calculating Colimits Compositionally. Recent years witnessed a renewed interest in exploring the dichotomy between the algebraic and the graphical presentations of a system, a topics to which Ugo has also contributed. Along these lines, the paper gives an algebraic description for finite colimits in a category based on the cospan construction, whence the graphical counterpart.

Donald Sannella and Andrzej Tarlecki: Observability concepts in abstract data type specification, 30 years later. Last but not least, the paper is ideal for closing our overview, because it presents in a modern fashion the pioneering ideas of Ugo on abstract data type specification [43], commenting upon which we opened this contribution.

References

1. Baldan, P., Bruni, R., Montanari, U.: Pre-nets, read arcs and unfolding: A functorial presentation. In: Wirsing, M., Pattinson, D., Hennicker, R. (eds.) WADT 2003. LNCS, vol. 2755, pp. 145–164. Springer, Heidelberg (2003)
2. Baldan, P., Corradini, A., Montanari, U.: Unfolding and event structure semantics for graph grammars. In: Thomas, W. (ed.) ETAPS 1999 and FOSSACS 1999. LNCS, vol. 1578, pp. 73–89. Springer, Heidelberg (1999)
3. Baldan, P., Corradini, A., Montanari, U.: Contextual Petri nets, asymmetric event structures, and processes. Inform. and Comput. 171(1), 1–49 (2001)
4. Baldan, P., Corradini, A., Montanari, U.: Relating SPO and DPO graph rewriting with Petri nets having read, inhibitor and reset arcs. Elect. Notes in Th. Comput. Sci. 127(2), 5–28 (2005)
5. Baldan, P., Corradini, A., Montanari, U., Ribeiro, L.: Unfolding semantics of graph transformation. Inform. and Comput. 205(5), 733–782 (2007)
6. Bonchi, F., König, B., Montanari, U.: Saturated semantics for reactive systems. In: Proc. of LICS 2006, pp. 69–80. IEEE Computer Society Press, Los Alamitos (2006)
7. Bruni, R., Butler, M.J., Ferreira, C., Hoare, C.A.R., Melgratti, H.C., Montanari, U.: Comparing two approaches to compensable flow composition. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 383–397. Springer, Heidelberg (2005)
8. Bruni, R., de Frutos-Escrig, D., Martí-Oliet, N., Montanari, U.: Bisimilarity congruences for open terms and term graphs via tile logic. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 259–274. Springer, Heidelberg (2000)
9. Bruni, R., Gadducci, F., Montanari, U.: Normal forms for algebras of connections. Theoret. Comput. Sci. 286(2), 247–292 (2002)
10. Bruni, R., Gadducci, F., Montanari, U., Sobocinski, P.: Deriving weak bisimulation congruences from reduction systems. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 293–307. Springer, Heidelberg (2005)
11. Bruni, R., Lanese, I., Montanari, U.: A basic algebra of stateless connectors. Theoret. Comput. Sci. 366(1-2), 98–120 (2006)
12. Bruni, R., Melgratti, H.C., Montanari, U.: Extending the zero-safe approach to coloured, reconfigurable and dynamic nets. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) ACPN 2003. LNCS, vol. 3098, pp. 291–327. Springer, Heidelberg (2004)
13. Bruni, R., Melgratti, H.C., Montanari, U.: Nested commits for mobile calculi: Extending join. In: Proc. of IFIP TCS 2004, pp. 563–576. Kluwer Academic Publishers, Dordrecht (2004)
14. Bruni, R., Melgratti, H.C., Montanari, U.: Event structure semantics for nominal calculi. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 295–309. Springer, Heidelberg (2006)
15. Bruni, R., Melgratti, H.C., Montanari, U.: Event structure semantics for dynamic graph grammars. In: Proc. of PNGT 2006. Elect. Communic. of the EASST, vol. 2, EASST (2007)
16. Bruni, R., Meseguer, J., Montanari, U.: Symmetric monoidal and cartesian double categories as a semantic framework for tile logic. Math. Struct. in Comput. Sci. 12(1), 53–90 (2002)
17. Bruni, R., Meseguer, J., Montanari, U., Sassone, V.: Functorial models for Petri nets. Inform. and Comput. 170(2), 207–236 (2001)

18. Bruni, R., Meseguer, J., Montanari, U., Sassone, V.: Algebraic theories for contextual prenets. In: Blundo, C., Laneve, C. (eds.) ICTCS 2003. LNCS, vol. 2841, pp. 256–270. Springer, Heidelberg (2003)
19. Bruni, R., Montanari, U.: Cartesian closed double categories, their lambda-notation, and the pi-calculus. In: Proc. of LICS 1999, pp. 246–265. IEEE Computer Society Press, Los Alamitos (1999)
20. Bruni, R., Montanari, U.: Zero-safe nets: Comparing the collective and individual token approaches. Inform. and Comput. 156(1-2), 46–89 (2000)
21. Bruni, R., Montanari, U.: Dynamic connectors for concurrency. Theoret. Comput. Sci. 281(1–2), 131–176 (2002)
22. Bruni, R., Montanari, U., Rossi, F.: An interactive semantics of logic programming. Theory and Practice of Logic Programming 1(6), 647–690 (2001)
23. Bruni, R., Montanari, U., Sassone, V.: Observational congruences for dynamically reconfigurable tile systems. Theoret. Comput. Sci. 335(2-3), 331–372 (2005)
24. Bruni, R., Sassone, V.: Algebraic models for contextual nets. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 175–186. Springer, Heidelberg (2000)
25. Buscemi, M.G., Montanari, U.: A compositional coalgebraic model of fusion calculus. J. Log. Algebr. Program 72(1), 78–97 (2007)
26. Castellani, I., Montanari, U.: Graph grammars for distributed systems. In: Ehrig, H., Nagl, M., Rozenberg, G. (eds.) Graph Grammars 1982. LNCS, vol. 153, pp. 20–38. Springer, Heidelberg (1983)
27. Coccia, M., Gadducci, F., Montanari, U.: GS-lambda theories: A syntax for higher-order graphs. Elect. Notes in Th. Comput. Sci. 69 (2002)
28. Corradini, A., Ehrig, H., Löwe, M., Montanari, U., Rossi, F.: An event structure semantics for safe graph grammars. In: Pro. of PROCOMET 1994. IFIP Transactions, vol. A-56, pp. 423–444. North-Holland, Amsterdam (1994)
29. Corradini, A., Heckel, R., Montanari, U.: Tile transition systems as structured coalgebras. In: Ciobanu, G., Păun, G. (eds.) FCT 1999. LNCS, vol. 1684, pp. 13–38. Springer, Heidelberg (1999)
30. Corradini, A., Montanari, U.: An algebraic semantics for structured transition systems and its application to logic programs. Theoret. Comput. Sci. 103, 51–106 (1992)
31. Corradini, A., Montanari, U., Rossi, F.: An abstract machine for concurrent modular systems: CHARM. Theoret. Comput. Sci. 122(1–2), 165–200 (1994)
32. Darondeau, P., Degano, P.: Causal trees. In: Ronchi Della Rocca, S., Ausiello, G., Dezani-Ciancaglini, M. (eds.) ICALP 1989. LNCS, vol. 372, pp. 234–248. Springer, Heidelberg (1989)
33. Degano, P., Meseguer, J., Montanari, U.: Axiomatizing the algebra of net computations and processes. Acta Inform. 33(7), 641–667 (1996)
34. Degano, P., Montanari, U.: A model for distributed systems based on graph rewriting. Journal of the ACM 34(2), 411–449 (1987)
35. Ferrari, G.L., Gnesi, S., Montanari, U., Pistore, M.: A model-checking verification environment for mobile processes. ACM Transactions on Software Engineering and Methodology 12(4), 440–473 (2003)
36. Ferrari, G.L., Hirsch, D., Lanese, I., Montanari, U., Tuosto, E.: Synchronised hyperedge replacement as a model for service oriented computing. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2005. LNCS, vol. 4111, pp. 22–43. Springer, Heidelberg (2006)
37. Ferrari, G.L., Montanari, U.: Parameterized structured operational semantics. Fundam. Inform 34(1-2), 1–31 (1998)
38. Ferrari, G.L., Montanari, U.: Tile formats for located and mobile systems. Inform. and Comput. 156(1/2), 173–235 (2000)

39. Ferrari, G.L., Montanari, U., Mowbray, M.: Structured transition systems with parametric observations: observational congruences and minimal realizations. Math. Struct. in Comput. Sci. 7(3), 241–282 (1997)
40. Ferrari, G.L., Montanari, U., Tuosto, E.: A LTS semantics of ambients via graph synchronization with mobility. In: Restivo, A., Ronchi Della Rocca, S., Roversi, L. (eds.) ICTCS 2001. LNCS, vol. 2202, pp. 1–16. Springer, Heidelberg (2001)
41. Gadducci, F., Montanari, U.: The tile model. In: Proof, Language and Interaction: Essays in Honour of Robin Milner, pp. 133–166. MIT Press, Cambridge (2000)
42. Gadducci, F., Montanari, U.: Comparing logics for rewriting: rewriting logic, action calculi and tile logic. Theoret. Comput. Sci. 285(2), 319–358 (2002)
43. Giarratana, V., Gimona, F., Montanari, U.: Observability concepts in abstract data type specifications. In: Mazurkiewicz, A. (ed.) MFCS 1976. LNCS, vol. 45, pp. 576–587. Springer, Heidelberg (1976)
44. Gorrieri, R., Marchetti, S., Montanari, U.: A2CCS: Atomic actions for CCS. Theoret. Comput. Sci. 72(2&3), 203–223 (1990)
45. Hirsch, D., Inverardi, P., Montanari, U.: Reconfiguration of software architecture styles with name mobility. In: Porto, A., Roman, G.-C. (eds.) COORDINATION 2000. LNCS, vol. 1906, pp. 148–163. Springer, Heidelberg (2000)
46. Hirsch, D., Montanari, U.: Synchronized hyperedge replacement with name mobility. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 121–136. Springer, Heidelberg (2001)
47. König, B., Montanari, U.: Observational equivalence for synchronized graph rewriting with mobility. In: Kobayashi, N., Pierce, B.C. (eds.) TACS 2001. LNCS, vol. 2215, pp. 145–164. Springer, Heidelberg (2001)
48. Lanese, I., Montanari, U.: Hoare vs Milner: Comparing synchronizations in a graphical framework with mobility. Elect. Notes in Th. Comput. Sci. 154(2), 55–72 (2006)
49. Laneve, C., Montanari, U.: Axiomatizing permutation equivalence. Math. Struct. in Comput. Sci. 6(3), 219–249 (1996)
50. Martelli, A., Montanari, U.: An efficient unification algorithm. ACM Transactions on Programming Languages and Systems 4, 258–282 (1982)
51. Meseguer, J.: Conditional rewriting logic as a unified model of concurrency. Theoret. Comput. Sci. 96, 73–155 (1992)
52. Meseguer, J., Montanari, U.: Petri nets are monoids. Inform. and Comput. 88, 105–155 (1990)
53. Meseguer, J., Montanari, U.: Mapping tile logic into rewriting logic. In: Parisi-Presicce, F. (ed.) WADT 1997. LNCS, vol. 1376, pp. 62–91. Springer, Heidelberg (1998)
54. Meseguer, J., Montanari, U., Sassone, V.: Process versus unfolding semantics for place/transition Petri nets. Theoretical Computer Science 153(1–2), 171–210 (1996)
55. Meseguer, J., Montanari, U., Sassone, V.: On the semantics of place/transition Petri nets. Math. Struct. in Comput. Sci. 7(4), 359–397 (1997)
56. Meseguer, J., Montanari, U., Sassone, V.: Representation theorems for Petri nets. In: Freksa, C., Jantzen, M., Valk, R. (eds.) Foundations of Computer Science. LNCS, vol. 1337, pp. 239–249. Springer, Heidelberg (1997)
57. Montanari, U., Pistore, M.: Structured coalgebras and minimal HD-automata for the pi-calculus. Theoret. Comput. Sci. 340(3), 539–576 (2005)
58. Rossi, F., Montanari, U.: Graph rewriting, constraint solving and tiles for coordinating distributed systems. Applied Categorical Structures 7(4), 333–370 (1999)