Electronic Notes in Theoretical Computer Science 15 (1998)
URL: http://www.elsevier.nl/locate/entcs/volume15.html 10 pages

A Logic for Modular Descriptions of Asynchronous and Synchronized Concurrent Systems [1]

Roberto Bruni

Computer Science Department, University of Pisa
Corso Italia 40, 56125 Pisa, Italy
Email: bruni@di.unipi.it

Abstract

Tile logic is a framework to reason about the dynamic evolution of concurrent systems in a modular way, and it extends *rewriting logic* (in the unconditional case) by rewriting synchronization and side effects. The subject of this dissertation concerns some interesting tile models of computation such that the mathematical structures representing *configurations* (i.e., system states) and *effects* (i.e., observable actions) have in common some auxiliary structure (e.g., for tupling, projecting, etc.). In particular, two such logics (called *process* and *term tile logic* respectively) are fully discussed, where *net-process-like* and usual *term* structures are employed. The categorical models for the two logics are introduced in terms of suitable classes of double categories. Then, the new model theory of *2EVH-categories* is proposed to relate the categorical models of tile logic and rewriting logic. This is particularly important, because rewriting logic is the semantic basis of several language implementation efforts (Cafe, ELAN, Maude), and therefore a conservative mapping of tile logic back into rewriting logic can be useful to get executable specifications of tile systems. The new model theory required to relate the two logics is presented in *partial membership equational logic*, a recently developed framework, which is particularly suitable to deal with categorical structures. The comparison yields a correct rewriting implementation of tile logic that uses a meta-layer to make sure that only tile proofs are performed. Exploiting the *reflective* capabilities of the Maude language, such meta-layer is specified in terms of *internal strategies*. Some detailed examples illustrating the implementation of tile systems for interesting *concurrent process calculi* conclude the presentation.

[1] Research supported by Esprit Working Groups *CONFER2* and *COORDINA*. Research carried out in part while the author was visiting at Computer Science Laboratory, SRI International, Menlo Park, CA, USA.

Overview

The present dissertation is about the theoretical and implementation aspects of a formalism, namely the *tile model* [20,22], for modular descriptions of concurrent system behaviours. The more practical part concerns the realization in Maude [11] (a reflective language recently developed at SRI International and based on *rewriting logic* [28,29]) of general meta-strategies for computing tile specifications. The theoretical part provides a categorical semantic setting that is also important to prove the correctness of the implementation. This presentation is based on a joint work with José Meseguer and Ugo Montanari [8], and will constitute most part of my PhD Thesis [5].

The notion of tile model relies on the use of a set of rules (called *tiles*) to define the behaviour of certain basic modules, which may interact through their interfaces (a module can be imagined just as a partially specified state of the system). Then, the behaviour of a whole system is defined as a coordinated evolution of its submodules. Graphically, a tile has the form

$$\begin{tikzcd} \circ \arrow[r, "s"] \arrow[d, "a"'] & \circ \arrow[d, "b"] \\ \circ \arrow[r, "s'"'] & \circ \end{tikzcd}$$

and textually it is written $s \xrightarrow[b]{a} s'$, stating that the *initial configuration* s of the system can evolve to the *final configuration* s' producing an *effect* b, which can be observed by the rest of the system. However, such a step is allowed only if the subcomponents of s (which is in general an *open* configuration) evolve to the subcomponents of s', producing the effect a, which acts as the *trigger* of the rule. The vertices $\circ$ of the tile are called *interfaces* (each configuration has both an *input* and an *output* interface). Tiles can be composed horizontally (through side effects), vertically (computational evolutions of a configuration), and in parallel (concurrent steps) to generate larger rules. It is evident that the tile model extends rewriting logic (in the unconditional case), taking into account side effects and rewriting synchronization (in unconditional rewriting systems, both triggers and effects are just identities; therefore, rewriting steps may be applied freely, i.e., without interacting with the rest of the system), and can be naturally equipped with observational equivalences and congruences based on effects. These aspects are very important, for example, when modelling process algebras via a rewrite system, because the behaviour of most process algebras depends on the interaction between agents and “the rest of the world”.

By analogy with rewriting logic, where a logic theory is associated to a term rewriting system in such a way that each concurrent computation represents a *sequent* entailed by the theory, the tile model also comes equipped with a purely logical presentation [22], where tiles are just considered as special (decorated) sequents subject to certain inference rules. Given a tile system, the associated *tile logic* is obtained by adding some auxiliary tiles and then freely composing in all possible ways (i.e., horizontally, vertically and in parallel) both auxiliary and basic tiles. Auxiliary tiles may be necessary to represent consistent rearrangements of the interfaces. Their role will be made more precise in the following part of the present overview.

In the recent literature, tiles have been used with success for modelling in detail several classes of applications, extending the SOS specification approach [34] to *open* and *heterogeneous* systems. Applications range from *coordination languages*, where triggers and effects naturally represent coordination protocols, to *mobile processes*, where free and bound names, and name extrusion must be handled.

In [33], a simple coordination model based on graph rewriting and synchronization is presented, using a class of tiles whose configurations are terms and whose effects are the elements of the free monoid over a set of basic observations. The hard computational problem of tile synchronization (which is in general NP-complete) is reduced to a distributed version of constraint solving, for which effective approaches exist. As an another example, tile models for most process algebras [22] have process terms as configurations, and elements of the free monoid of actions (which are unary symbols) as observations.

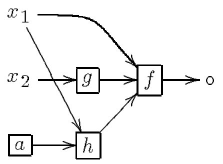
The simplest possible interpretation of structured configurations and observations is considered in [6,7], and consists of P/T net markings. As an important result, horizontal composition in the tile model yields a notion of *transition synchronization*, an important feature for compositionality, which is missing in ordinary nets (where only *token* synchronization is provided), and usually achieved through complex constructions.

Since models of computation based on the notion of free and bound names are widespread, the notion of *name sharing* is essential for several applications, ranging from logic programming, λ-calculus and process algebra with restriction (or name hiding mechanisms) to mobile processes (where local names may be communicated to the external world, thus becoming global names). We can think of names as links to communication channels, or to objects, or to locations, or to remote shared resources, or, also, to some *cause* in the event history of the system. In general, names can be freely α-converted, because the only important information they offer is the sharing of common resources. A main advantage of tiles is that data structures for configurations and effects are not restricted to be syntactic terms. A small variation over ordinary terms which has proved very expressive is *term graphs* [15]: they are a reference-oriented generalization of the ordinary (value-oriented) notion of term, where the sharing of subterms can be specified also for closed (i.e., without variables) term graphs (terms can share variables, but shared subterms of closed terms can be freely copied, always yielding an equivalent term).

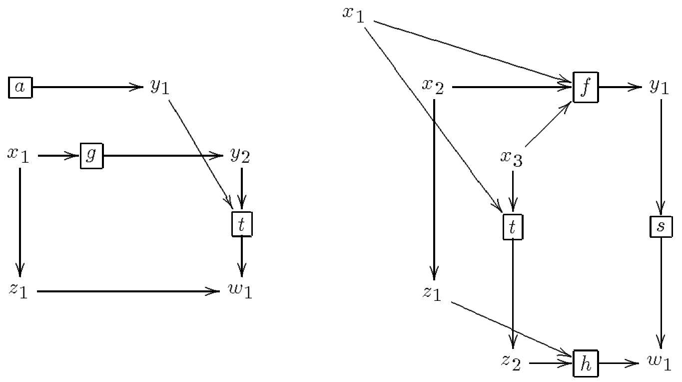
Such structures have been shown useful in [17] to define a tile model for the (asynchronous) π-calculus [32] (one of the most studied mobile calculi), and in [18] to represent both the operational and the abstract semantics of CCS with locations [3] within the tile model. In both cases, the general notion of *tile bisimilarity* [22] is employed to quotient out configurations, thus recovering the ordinary abstract semantics. Name extrusion and explicit handling make the transition system for the π-calculus infinite branching and require special notions of bisimulation. However, extrusion does not present any problem and can be specified using finite branching if one uses tiles whose configurations and effects are term graphs. Indeed, term graphs can handle names as wires, i.e., every sharing connection must be defined and explicit naming is not necessary. In the case of CCS with locations, sharing is used within configurations for modeling the parallel composition operator $\_ \mid \_$ of the process algebra, which, in this context, means sharing *the same location*. Within observations, sharing means that two events share *the same cause*, or, equivalently, that the same location has two different sublocations.

The main issue of this presentation is the study of a conservative embedding of tile logic into rewriting logic, focusing on the case in which configurations and effects rely on common auxiliary structures (e.g., for tupling, projecting or permuting interfaces). This is useful because there exist several languages based on rewriting logic (Cafe [19], ELAN [2], Maude [11]), and the implementation of such a mapping immediately supports the execution of tile specifications.

As illustrated above, the notions of configuration and effect are very general here: the only requirement is that they come equipped with operations of parallel and sequential composition. To give a formal definition of auxiliary structure we can assume that they form two *monoidal categories* having the same class of objects. In particular, a general and convenient categorical characterization of configurations and effects can be given in terms of *algebraic theories* [25,26,24]. The free algebraic theory associated to a (one-sorted) signature Σ is called the Lawvere theory for Σ, and is denoted by $\mathcal{L}_\Sigma$: the objects are underlined natural numbers, the arrows from $\underline{m}$ to $\underline{n}$ are in a one-to-one correspondence with n-tuples of terms of the free Σ-algebra with (at most) m variables, and composition is term substitution. In a certain sense, a Lawvere theory is just an alternative presentation of a signature, because the additional structure (for tupling, projecting and permuting the elements of a tuple) is generated in a completely free way: only the operators of the signature contain information, whereas the other constructors add nothing but auxiliary structure. From this point of view, the use of a *wires and boxes* notation turns out to be very useful for a visual and intuitive understanding of the role played by the auxiliary structure: variables are represented by wires (we assume an implicit total order of the variables involved) and the operators of the signature are denoted by boxes labelled with the name of the operators. For instance, the term $f(x_1, g(x_2), h(x_1, a))$ over the signature $\Sigma = \{a : 0 \longrightarrow 1, g : 1 \longrightarrow 1, h : 2 \longrightarrow 1, f : 3 \longrightarrow 1\}$ and variables $x_1 < x_2$ admits the following graphical representation:



It should be obvious that wire duplications (e.g., of x_1) and crossing of wires (e.g., of x_2 and a copy of x_1) are auxiliary, in the sense that they belong to any wires and boxes model, independently from the underlying signature. It follows that, if we use the wires and boxes notation for configurations and effects, then this kind of operations (e.g., rearrangements of wires) belongs to both dimensions (i.e., they are shared). Moreover, consistent rearrangements of wires on both dimensions do not change the meaning of a rule, but only its interface. To illustrate this point, let us consider a simple tile system, where the above signature Σ is the signature of configurations, the signature of effects is $\Sigma' = \{s : 1 \longrightarrow 1, t : 2 \longrightarrow 1\}$, and the basic tiles are:



Then, one may expect that the configuration $f(a, x_1, g(x_2))$ is able to evolve to $h(x_1, x_2)$, producing an effect s (as a result of the horizontal composition, or synchronization, of the two tiles). However, it is impossible to compose the tiles in the obvious way without rearranging the interfaces, because the arguments of the trigger t of the second tile are separated by a variable in the initial input interface (notice the crossing of wires), while the first tile applies the effect to adjacent arguments (notice that it is always possible to put an idle component in parallel with the first tile to model the second argument of f). Thus we have the following *naïve* characterization of auxiliary tiles:

> Auxiliary tiles coincide with the consistent rearrangements of interfaces in both dimensions, where consistency means that the composition of the wire transformations induced by the initial configuration and the effect of the tile is equivalent to the composition of the wire transformations due to the trigger and the final configuration.

Algebraic theories provide a clear mathematical representation of auxiliary constructors as suitable natural transformations, whose components are called *symmetries*, *duplicators*, and *dischargers*. This result will be very useful to relate our naïve definition with a more formal definition.

Lawvere theories introduce a very general notion of *model* (i.e., a chosen functor from $\mathcal{L}_\Sigma$ to a cartesian category with chosen products) and *model morphism* (i.e., a natural transformation between two models). This fact has been well-exploited in the categorical semantics of rewriting systems. In fact, in the field of term rewriting, the states are terms over a certain signature (i.e., arrows of the associated Lawvere theory), and rewriting steps are transitions between two terms (with variables). It has been shown in [28] that a rewriting theory $\mathcal{R}$ yields a cartesian *2-category*[2] $\mathcal{L}_\mathcal{R}$, which does for $\mathcal{R}$

[2] A 2-category [23,27] is a category $\mathcal{C}$ such that, for any two objects a, b, the class $\mathcal{C}[a, b]$ of arrows from a to b in $\mathcal{C}$, forms a (vertical) category. The arrows of these hom-categories are

what a Lawvere theory does for a signature (i.e., models can be defined as 2-product-preserving 2-functors). Gadducci and Montanari pointed out in [21] that if also side-effects are to be taken into consideration during the rewriting process, then *double categories* [16,1,23] should be considered as a natural model. A double category can be informally described as the superposition of a horizontal and a vertical category of *cells*, the former defining effect propagations, and the latter describing state evolutions. Then, in the same way as the term algebra is freely generated by a signature, and the initial model of rewriting logic is freely generated from the rules of the rewriting system, the tiles freely generate a (monoidal) double category which constitutes the natural operational characterization [3] in the spirit of initial model semantics.

Two main interesting cases of shared auxiliary structures are considered here, yielding the notions of *process tile logic* and *term tile logic*:

- *Flat* (i.e., any two sequents having the same "border" are identified, thus no emphasis is given upon the axiomatization of logic proofs) versions of process tile logic have been shown to be especially useful for defining compositional models of computation of mobile calculi, and causal and located concurrent systems [17,18]. The auxiliary tiles of process tile logic express consistent permutations of interfaces along the horizontal and vertical structures.
- Term tile logic represents the obvious extension of term rewriting logic. Connections between the two logics are particularly interesting because in both logics the underlying cartesian category structure manifests itself at the level of syntax, allowing the use of the standard term notation with term substitution as composition. The auxiliary tiles of term tile logic allow consistent permutations of interfaces along the horizontal and vertical structures (as for process tile logic), consistent free copying, and consistent projections on subcomponents.

The natural semantics of process and term tile logics are given in terms of suitable classes of double categories whose equational axioms identify intuitively equivalent tile computations. For this purpose, the notions of *symmetric strict monoidal double category* and *cartesian double category with consistently chosen products* are introduced. As far as we know these definitions are original, because previous attempts (based on internal constructions) for analogous notions have led to asymmetric models, where the auxiliary structure (i.e., symmetries, duplicators, and dischargers) is fully exploited in one dimension only. We believe that this should not be the case, both conceptually and for the kind of applications we have in mind; therefore we developed an alternative approach, following the idea of *hyper-transformations* [16] for many-fold categories, and exploiting the results for double categories. In par-

called *cells* and satisfy particular composition properties. For example, the category **Cat** of categories and functors is a 2-category ($\mathbf{Cat}[\mathcal{C},\mathcal{C}']$ is the category having the functors from $\mathcal{C}$ to $\mathcal{C}'$ as objects, and the natural transformations between such functors as arrows).

[3] The tiles are cells, the contexts are arrows of the 1-horizontal category, the side-effects are the arrows of the vertical 1-category, and 0-objects model connections between the somehow syntactic horizontal category and the dynamic vertical evolution.

ticular, we define the notion of *generalized transformations* (behaving the same in both dimensions), and assert the coherence of the two ways of transforming the structure. Then, we instantiate the definition to the special cases of symmetries, duplicators, and dischargers, in a similar way as it happens for the 1-dimensional case. Moreover, by doing that, we give evidence for the usefulness of axiomatizing the resulting double categories, thus allowing for the definition of more significant models than the flat ones. Actually such models could also take into account the structure of proofs. This approach motivates the following formal characterization of auxiliary tiles:

> Auxiliary tiles for process and term tile logic are suitable generalized transformations respecting some coherence equations, where coherence means that they are uniquely defined.

The comparison between tile logic and rewriting logic is carried out by embedding their corresponding categorical models in a recently developed, more general framework, called *partial membership equational logic* (PMEqtl) [30,4]. PMEqtl is particularly suitable for the modelling and the embedding of categorical structures, firstly because the sequential composition of arrows is a partial operation (it is defined if and only if the target of the first argument is equal to the source of the second argument), and secondly because membership predicates over a poset of sorts allow modelling the objects as a subset of the arrows and arrows as a subset of cells (as it is usually done in category theory). Moreover, the *tensor product construction* illustrated in [31] can be easily formulated in PMEqtl, yielding a convenient definition of monoidal double categories as the tensor product of the theory of categories (twice) with the theory of monoids. To accomplish the comparison, we define an extended version of 2-category, called *2EVH-category*, that provides a systematic connection between models of tile logic and of rewriting logic. The idea is to "stretch" double cells into ordinary 2-cells as pictured below, mantaining the capability to distinguish between configurations and effects, whereas the auxiliary structure becomes shared, i.e., it belongs to both classes.

$$\begin{tikzcd} \circ \arrow[r, "s"] \arrow[d, "a"'] & \circ \arrow[d, "b"] \\ \circ \arrow[r, "s'"'] & \circ \end{tikzcd} \qquad \begin{tikzcd} & \circ \arrow[dr, "b"] & \\ \circ \arrow[ur, "s"] \arrow[dr, "a"'] & \Downarrow & \circ \\ & \circ \arrow[ur, "s'"'] & \end{tikzcd}$$

Doing this, 2EVH-categories are able to *simulate* – in the sense that the algebraic structure of the original double categories is recoverable in terms of operations on 2-cells – the structure of double categories, where both the horizontal and vertical 1-categories share some non-trivial structure other than objects. In this translation we must be careful about two issues, namely, the possible identification of distinct double cells, and the possible existence of 2-cells having correct horizontal-vertical partition of the source and vertical-horizontal partition of the target, but which do not represent any double cell. From the facts that: (1) each arrow of a 2-category can be viewed as an identity 2-cell, (2) each auxiliary operator is a shared arrow, and (3) auxiliary tiles are consistent (in the sense that the composition of s with b is equivalent to the composition of a with s'), it follows that 2EVH-categories allow for a third characterization of auxiliary tiles:

> Auxiliary tiles coincide with the possible square-shaped decompositions of the identity 2-cells associated to auxiliary constructors.

An important result states the equivalence of the three different definitions of auxiliary tiles that we have sketched in this overview.

Though the results are very satisfying from a theoretical perspective, they cannot be applied directly to rewriting implementations of tile systems, because we are interested only in correct tile computations. Indeed, we need suitable meta-strategies to control the possible nondeterminism contained in a tile specification and also in its translation. To overcome this difficulty, we make use of the *reflective* capabilities of Maude to define suitable *internal strategies* [14,12,13,10], that can help the user to control the rewritings and to collect (some of) the possible (correct) results. The key point is that the internal strategies defined here for simulating tile systems can also be thought of as general meta-strategies for nondeterministic rewriting systems. We have experimented with Maude some executable tile specifications for finite CCS and for located CCS, and have successfully developed and applied general internal strategies to analyze tile computations (see also [9]).

Acknowledgements

First of all I wish to thank my advisor Ugo Montanari, who influenced the direction of research in my thesis with many original ideas and contributions arising from his impressive knowledge of most of the different fields of Computer Science. The other person who has mostly contributed to the development of my thesis is José Meseguer. It has been a pleasure to work with him during the months I spent in California at SRI, and although our collaboration started very recently (Autumn 1997), I think that it has been very fruitful and promising.

I want to thank Fabio Gadducci for introducing me to the "world of tiles" and for many discussion on related topics, and Vladimiro Sassone that helped me in some preliminary investigations on the notion of symmetric double categories (at the end of 1996). I also thank Narciso Martí-Oliet for his precious suggestions and comments on a preliminary version of the T.R. [8], which have improved a lot the exposition, and Manuel Clavel for his help in the area of strategies. The final thanks are for the Organizers of the WRLA98 conference who gave me this opportunity to present my work.

References

[1] A. Bastiani and C. Ehresmann. Multiple Functors I: Limits Relative to Double Categories. *Cahiers de Topologie et Géométrie Diff.* **15**:3, 545-621 (1974).

[2] P. Borovanský, C. Kirchner, H. Kirchner, P.-E. Moreau, and M. Vittek. ELAN: A logical framework based on computational systems. In: J. Meseguer, Ed.,

Proc. First International Workshop on Rewriting Logic and its Applications, *ENTCS* **4** (1996).

[3] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing Localities. *TCS* **114**, 31–61 (1993).

[4] A. Bouhoula, J.P. Jouannaud, and J. Meseguer. Specification and proof in membership equational logic. In M. Bidoit and M. Dauchet, Eds., *Proceedings of TAPSOFT'97*, *LNCS* **1214**, 67–92 (1997).

[5] R. Bruni. PhD Thesis, Department of Computer Science, University of Pisa, forthcoming.

[6] R. Bruni and U. Montanari. Zero-Safe Nets, or Transition Synchronization Made Simple. In: C. Palamidessi, J. Parrow, Eds. *Proceedings of EXPRESS'97*, *ENTCS* **7** (1997).

[7] R. Bruni and U. Montanari, Zero-Safe Nets: The Individual Token Approach. In: F. Parisi-Presicce, Ed., *Proc. 12th Workshop on Algebraic Development Techniques*, *LNCS* **1376**, 122–140 (1998).

[8] R. Bruni, J. Meseguer, and U. Montanari. Process and Term Tile Logic. Technical Report SRI-CSL-98-06, SRI International (July 1998). Also Technical Report TR-98-09, Department of Computer Science, University of Pisa (1998).

[9] R. Bruni, J. Meseguer, and U. Montanari. Internal Strategies in a Rewriting Implementation of Tile Systems. This volume.

[10] M.G. Clavel. *Reflection in General Logics and in Rewriting Logic with Applications to the Maude Language*. PhD Thesis, Univ. de Navarra (1998).

[11] M.G. Clavel, F. Duran, S. Eker, P. Lincoln, and J. Meseguer. *An Introduction to Maude (Beta Version)*. SRI International, Menlo Park, CA, USA (1998).

[12] M.G. Clavel, and J. Meseguer. Reflection and Strategies in Rewriting Logic. In: J. Meseguer, Ed., *Proc. First International Workshop on Rewriting Logic and its Applications*, *ENTCS* **4** (1996).

[13] M.G. Clavel, and J. Meseguer. Axiomatizing Reflective Logics and Languages. In: G. Kiczales, Ed., *Proc. Reflection'96*. San Francisco, USA, 263–288 (1996).

[14] M.G. Clavel, and J. Meseguer. Internal Strategies in a Reflective Logic. In: B. Gramlich, and H. Kirchner, Eds., *Proc. of the CADE-14 Workshop on Strategies in Automated Deduction*, Townsville, Australia, 1–12 (1997).

[15] A. Corradini and F. Gadducci. A 2-categorical Presentation of Term Graph Rewriting. In: E. Moggi, G. Rosolini, Eds., *Proc. CTCS'97*, *LNCS* **1290**, 87–105 (1997).

[16] C. Ehresmann. Catégories Structurées: I and II, *Ann. Éc. Norm. Sup.* **80**, 349–426, Paris (1963); III, *Topo. et Géo. Diff.* **V**, Paris (1963).

[17] G.L. Ferrari and U. Montanari. A Tile-Based Coordination View of Asynchronous Pi-Calculus. In: I. Prívara, P. Ruzicka, Eds., *Mathematical Foundations of Computer Science 1997*, *LNCS* **1295**, 52–70 (1997).

[18] G.L. Ferrari and U. Montanari. Tiles for Concurrent and Located Calculi. In: C. Palamidessi, J. Parrow, Eds., *Proc. of EXPRESS'97.* ENTCS Vol.7, 1997.

[19] K. Futatsugi and T. Sawada. Cafe as an extensible specification environment. In *Proc. of the Kunming International CASE Symp.*, Kunming, China (1994).

[20] F. Gadducci. *On the Algebraic Approach to Concurrent Term Rewriting.* PhD Thesis TD-96-02. Department of Computer Science, University of Pisa (1996).

[21] F. Gadducci and U. Montanari. Enriched Categories as Models of Computations. In: *Proc. 5th Italian Conference on Theoretical Computer Science, ITCS'95*, World Scientific, 1–24 (1996).

[22] F. Gadducci and U. Montanari. The Tile Model. In: G. Plotkin, C. Stirling, M. Tofte, Eds., *Proof, Language and Interaction: Essays in Honour of Robin Milner*, MIT Press, to appear. Also Technical Report TR-96-27, Department of Computer Science, University of Pisa (1996).

[23] G.M. Kelly and R.H. Street. Review of the Elements of 2-categories. *Lecture Notes in Mathematics* **420**, 75–103 (1974).

[24] A. Kock and G.E. Reyes. Doctrines in Categorical Logic. In: John Barwise, Ed., *Handbook of Mathematical Logic*, North Holland, 283–313 (1977).

[25] F.W. Lawvere. Functorial Semantics of Algebraic Theories. In *Proc. National Academy of Science* **50**, 869–872 (1963).

[26] F.W. Lawvere. Some algebraic problems in the context of functorial semantics of algebraic theories. In *Proc. 2th Midwest Category Seminar. Lecture Notes in Mathematics* **61**, 41–61 (1968).

[27] S. MacLane. *Categories for the Working Mathematician.* Springer-Verlag (1971).

[28] J. Meseguer, *Rewriting as a Unified Model of Concurrency*, SRI Technical Report, CSL-90-02R (February 1990. Revised June 1990).

[29] J. Meseguer. Conditional Rewriting Logic as a Unified Model of Concurrency. *TCS* **96**, 73–155 (1992).

[30] J. Meseguer. Membership Equational Logic as a Logical Framework for Equational Specification. In: F. Parisi-Presicce, Ed., *Proc. 12th Workshop on Algebraic Development Techniques, LNCS* **1376**, 18–61 (1998).

[31] J. Meseguer and U. Montanari. Mapping Tile Logic into Rewriting Logic. In: F. Parisi-Presicce, Ed., *Proc. 12th Workshop on Algebraic Development Techniques, LNCS* **1376**, 62–91 (1998).

[32] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes (parts I and II). *Information and Computation* **100**, 1–77 (1992).

[33] U. Montanari and F. Rossi. Graph Rewriting, Constraint Solving and Tiles for Coordinating Distributed Systems. Draft submitted for publication.

[34] G. Plotkin. *A Structural Approach to Operational Semantics.* Technical Report DAIMI FN-19, Computer Science Department, Aarhus University (1981).