Rewriting Logic: Roadmap and Bibliography ⋆

Narciso Martí-Oliet [a] and José Meseguer [b]

[a] *Facultad de Matemáticas, Universidad Complutense, Madrid, Spain*
[b] *SRI International, Menlo Park, California, USA*

1 Introduction

The theory and applications of rewriting logic have been vigorously developed by researchers all over the world during the past eleven years. The attached bibliography includes more than three hundred papers related to rewriting logic that have been published so far. Three international workshops on rewriting logic have been held in the United States, France, and Japan [222,167,139], and a fourth will be held in Italy in 2002. Furthermore, as explained later in this roadmap, several language implementations and a variety of formal tools have also been developed and have been used in a wide range of applications.

Several snapshots of the state of rewriting logic research—some more global in scope, and others restricted to specific areas such as concurrency or object-based systems—have appeared so far [223,227,229,228]. The present survey is another such snapshot, but it is restricted on purpose on two counts: first in its length, which is relatively short; and second in discussing only work within the rewriting logic area. In particular, no attempt has been made to discuss work on related approaches serving as logical or semantic frameworks. In fact, it is not even a detailed survey of work in rewriting logic; instead, as its name suggests, it is a *roadmap* to help somebody interested in this area get the lay of the land, that is, a first general overview of the main concepts, results, and applications in what we think is a promising research area. In particular, the references cited in the roadmap do not try to be exhaustive, but only to give some illustrative examples. However, the bibliography itself contains all the relevant references that we are aware of at this time.

⋆ Supported by DARPA through Rome Laboratories Contract F30602-97-C-0312, by Office of Naval Research Contracts N00014-99-C-0198 and N00014-01-1-0837, and by National Science Foundation Grant CCR-9900334.

2 Basic Concepts

In rewriting logic [218] the basic axioms are rewrite rules of the form $t \longrightarrow t'$, with t and t' expressions in a given language. There are two complementary readings of a rewrite rule $t \longrightarrow t'$, one computational, and another logical:

- *computationally*, the rewrite rule $t \longrightarrow t'$ is interpreted as a *local transition* in a concurrent system; that is, t and t' describe patterns for *fragments* of the distributed state of a system, and the rule explains how a local concurrent transition can take place in such a system, changing the local state fragment from an instance of the pattern t to the corresponding instance of the pattern t'.
- *logically*, the rewrite rule $t \longrightarrow t'$ is interpreted as an *inference rule*, so that we can infer formulas of the form t' from formulas of the form t.

The computational and logical viewpoints are not exclusive: they complement each other and are, in some sense, in the eyes of the beholder. For example, a simple rewrite theory whose rewrite rules rewrite ground multisets built out of some constants by means of an associative and commutative multiset union operator, denoted, say, by $\otimes$, has an obvious computational reading as a (place/transition) Petri net; and an equally obvious logical reading as a tensor theory in propositional linear logic (for a discussion of these two readings see [211]).

A rewrite theory is a 4-tuple $\mathcal{R} = (\Sigma, E, L, R)$, where (Σ, E) is the equational theory *modulo which* we rewrite, L is a set of labels, and R is a set of labeled rules[1]. In the case of a Petri net, Σ consists of the binary multiset union operator $\otimes$ and one constant for each place in the net, E consists of the associativity and commutativity equations for multiset union, L is the set of labels of the net's transitions, and R is the set of transitions. Since we rewrite *modulo* the equations E, what are really rewritten are *equivalence classes* of terms modulo E. In the Petri net example this corresponds to the fact that each transition rewrites a (fragment of) the current multiset of places (graphically depicted as a "marking," with as many "tokens" in a place as its multiplicity) modulo the associativity and commutativity of multiset union.

As a consequence, the relevant *sentences*—that may or may not be provable by the above theory $\mathcal{R}$—are sequents of the form $[t]_E \longrightarrow [t']_E$, where t and t' are Σ-terms, possibly involving some variables, and $[t]_E$ denotes the equivalence class of the term t modulo the equations E. The *provable* sentences are exactly

[1] For simplicity we will assume that R consists of *unconditional* labeled rules of the form $l : t \longrightarrow t'$, but all we say extends naturally to conditional rules that may contain rewrites in their conditions [218].

those derivable by the following inference rules[2]:

(1) **Reflexivity**. For each $[t] \in T_{\Sigma,E}(X)$, $\overline{[t] \longrightarrow [t]}$.

(2) **Congruence**. For each $f \in \Sigma_n$, $n \in \mathbb{N}$,

$$\frac{[t_1] \longrightarrow [t_1'] \quad \ldots \quad [t_n] \longrightarrow [t_n']}{[f(t_1, \ldots, t_n)] \longrightarrow [f(t_1', \ldots, t_n')]}.$$

(3) **Replacement**. For each rule $l : [t(x_1, \ldots, x_n)] \longrightarrow [t'(x_1, \ldots, x_n)]$ in R,

$$\frac{[w_1] \longrightarrow [w_1'] \quad \ldots \quad [w_n] \longrightarrow [w_n']}{[t(\overline{w}/\overline{x})] \longrightarrow [t'(\overline{w'}/\overline{x})]}.$$

(4) **Transitivity**

$$\frac{[t_1] \longrightarrow [t_2] \qquad [t_2] \longrightarrow [t_3]}{[t_1] \longrightarrow [t_3]}.$$

Computationally, the provable sequents describe all the complex concurrent transitions of the system axiomatized by $\mathcal{R}$. Logically, they describe all the possible complex deductions from one formula to another in the logic axiomatized by $\mathcal{R}$.

Besides having an inference system, rewriting logic also has a *model theory* with natural computational and logical interpretations. Furthermore, each rewrite theory $\mathcal{R}$ has an *initial model* $\mathcal{T}_{\mathcal{R}}$ [218]. The idea is that we can decorate the provable sequents with *proof terms*, indicating how indeed they have been proved. Computationally, a proof term is a description of a, possibly complex, concurrent computation; logically, it is of course a description of a logical deduction. The question is then, when should two such proof terms be considered *equivalent* descriptions of the same computation/deduction? In the model $\mathcal{T}_{\mathcal{R}}$ this is answered by equating proof terms according to natural equivalence equations [218]. In this way we obtain a model $\mathcal{T}_{\mathcal{R}}$ with a category structure, where the objects are E-equivalence classes of ground Σ-terms, and the arrows are equivalence classes of proof terms. Identities are naturally associated with reflexivity proofs; and arrow compositions correspond to transitivity proofs. The computational and logical interpretations are then obvious, since a category is a structured transition system; and logical systems have been understood as categories since the early work of Lambek on deductive systems. The proof theory and model theory of rewriting logic are related by a *completeness theorem*, stating that a sequent is provable from $\mathcal{R}$ if and only if it is satisfied in all models of $\mathcal{R}$ [218].

[2] For simplicity we treat here *unsorted* (and unconditional) rewriting logic; but the logic is in fact parameterized by the choice of its underlying equational logic: unsorted, many-sorted, order-sorted, membership, etc.

Yet another very important property of rewriting logic is *reflection* [82,65,85]. Intuitively, a logic is reflective if it can represent its metalevel at the object level in a sound and coherent way. Specifically, rewriting logic can represent its own theories and their deductions by having a finitely presented rewrite theory $\mathcal{U}$ that is *universal*, in the sense that for any finitely presented rewrite theory $\mathcal{R}$ (including $\mathcal{U}$ itself) we have the following equivalence

$$\mathcal{R} \vdash t \to t' \;\;\Leftrightarrow\;\; \mathcal{U} \vdash \langle \overline{\mathcal{R}}, \overline{t} \rangle \to \langle \overline{\mathcal{R}}, \overline{t'} \rangle,$$

where $\overline{\mathcal{R}}$ and $\overline{t}$ are terms, of respective sorts `Module` and `Term`, representing $\mathcal{R}$ and t as data elements of $\mathcal{U}$. Since $\mathcal{U}$ is representable in itself, we can achieve a "reflective tower" with an arbitrary number of levels of reflection [65,66].

Reflection is a very powerful property: it allows defining rewriting strategies by means of metalevel theories that extend $\mathcal{U}$ and guide the application of the rules in a given object-level theory $\mathcal{R}$ [65]; it can be efficiently supported in a language implementation by means of *descent functions* [66]; it can be used to build a variety of theorem proving and theory transformation tools [65,77,78]; it can endow a rewriting logic language with powerful theory composition operations [121,116,118,125]; and it can be used to prove metalogical properties about families of theories in rewriting logic, and about other logics represented in the rewriting logic logical framework [11,79].

How should rewrite theories be executed in practice? First of all, in a general rewrite theory $\mathcal{R} = (\Sigma, E, L, R)$ the equations E can be arbitrary, and therefore, E-equality may be undecidable. Assuming that the equations E are unconditional, a general solution is to transform $\mathcal{R}$ into a rewrite theory $\mathcal{R}^\sharp = (\Sigma, \emptyset, L \cup L_E, R \cup E \cup E^{-1})$ in which we view the equations E as rules from left to right (E) and from right to left (E^{-1}), labeled by appropriate new labels L_E. In this way, we can reduce the problem of rewriting modulo E to the problem of standard rewriting, since we have the equivalence

$$\mathcal{R} \vdash [t] \to [t'] \;\;\Leftrightarrow\;\; \mathcal{R}^\sharp \vdash t \to t'.$$

In actual specification and programming practice we can do much better than this, because the equational theory (Σ, E) is typically decidable. A commonly occurring form for the decidable equational theory (Σ, E) is with $E = E' \cup A$, where A is a set of equational axioms for which we have a matching algorithm, and E' is a set of Church-Rosser and terminating equations *modulo* A. In these circumstances, a very attractive possibility is to transform $\mathcal{R} = (\Sigma, E' \cup A, L, R)$ into the theory $\mathcal{R}^\dagger = (\Sigma, A, L \cup L_{E'}, R \cup E')$. That is, we now view the equations E' as rules added to R, labeled with appropriate new labels $L_{E'}$. In this way, we reduce the problem of rewriting modulo E to the much simpler problem of rewriting modulo A, for which, by assumption, we have a matching algorithm. The question is, of course, under which conditions is this transformation complete, that is, under which conditions do we have an equivalence

$$\mathcal{R} \vdash [t]_E \to [t']_E \quad \Leftrightarrow \quad \mathcal{R}^\dagger \vdash [t]_A \to [t']_A.$$

Conditions guaranteeing this equivalence center around different variations on the notion of *coherence*, which is a form of "relative confluence" between equations and rules. Methods for checking coherence, or for achieving it by a process of "relative completion," have been proposed by Viry in several papers [314,315,318].

Even when the rewrite theory is coherent and the language implementation supports rewriting modulo A, executing rewrite theories is nontrivial, because the rules R in general are neither Church-Rosser nor terminating. Furthermore, some rules in R may have *additional variables* on their righthand sides, yet another source of nondeterminism. For this reason, sequential implementations of rewriting logic typically support *rewriting strategies* that let the user specify how the rules should be applied [169,82,22,83,17,319,65]. Such strategies can be defined in metalevel theories by reflection, as already indicated, or they may be part of a strategy language supported by a language implementation. However, one should not forget that rewriting logic is an *intrinsically concurrent formalism*, that can be used directly for concurrent and distributed programming (see for example [238,202,120]). Therefore, whereas in a sequential implementation we are *simulating* a concurrent execution, and need a strategy to choose a particular interleaving computation, in a truly concurrent execution nondeterminism is a fact of life, and we may care much less about how rules are applied, and be much less able to control their application in practice. We may in fact allow many different computations, while still imposing some weaker requirements such as different forms of fairness.

3 Rewriting Logic and Formal Methods

The fact that, under reasonable assumptions, rewriting logic specifications are executable allows us to have a flexible range of *increasingly stronger formal methods*, to which a system specification can be subjected. Only after less costly and "lighter" methods have been used, it is meaningful and worthwhile to invest effort on "heavier" and costlier methods. A rewriting logic language implementation, together with an associated environment of formal tools, can be used to support the following, increasingly stronger methods [74]: (1) formal specification, (2) execution of the specification, (3) model-checking analysis, (4) narrowing analysis, and (5) formal proof.

Executability, combined with program transformation and compilation techniques, has yet another key advantage, namely, that rewriting logic specifications validated by the above formal methods can then be directly transformed and compiled for efficient execution. In fact, the state of the art in rewriting logic language implementations (see Section 6) suggests that for many applications the implementations thus obtained, besides being correct by construction, can compete in efficiency with implementations developed in conventional languages.

The above methodology should be supported by formal tools. First of all, a reflective rewriting logic implementation can directly support methods 1–3, and can also be used as a *reflective metatool* to develop other formal tools for methods 3–5. Maude has been used in exactly this way [78,77,262] to build tools such as an inductive theorem prover; a tool to check the Church-Rosser property, coherence, and termination, and to perform Knuth-Bendix completion; and a tool to specify, analyze and model check real-time specifications [267,262]. Some of the above tools have also been integrated within the formal tool environment of CafeOBJ [142]. Similarly, as further discussed in Section 5, both ELAN and Maude have been used to develop a wide variety of formal tools and automated deduction algorithms, based on quite different logics.

Rewriting logic is primarily a logic *of* change in which the deduction directly corresponds to the change [211], as opposed to a logic to talk *about* change in a more indirect and global manner, such as the different variants of modal and temporal logic. Such logics regard a system as a mathematical model—typically some kind of Kripke structure—about which they then make assertions about its global properties, such as safety or liveness properties. Both levels of description and analysis are useful in their own right; in fact, they complement each other: one can use both logics in combination to prove system properties.

The integration of these two logical levels is usually straightforward, because both logics are talking about essentially the same mathematical model. In fact, the initial model $\mathcal{T}_{\mathcal{R}}$ of a rewrite theory $\mathcal{R}$ is a category with algebraic structure, where the objects correspond to system states, and the arrows correspond to concurrent system transitions. Therefore, $\mathcal{T}_{\mathcal{R}}$ can be regarded as a Kripke structure whose transitions are labeled by the arrows of the category. A variety of different modal or temporal logics can then be chosen to make assertions about such a Kripke structure, or about a closely-related structure obtained from it, such as, for example, the extension $\mathcal{T}_{\mathcal{R}}^{\infty}$ of $\mathcal{T}_{\mathcal{R}}$ to infinite computations.

The investigation of suitable specification logics having a good integration with rewriting logic is an active area of research. In the choice of such a specification logic there are different tradeoffs between, for example, generality, expressiveness, and amenability to different deductive and/or model-checking techniques. Two general proposals for modal logics for reasoning about general rewrite theories are those of Fiadeiro et al. in [136], and the coalgebraic approach of Pattinson [272]. But since object-oriented systems constitute a particularly wide and important application area, modal or temporal logics that provide explicit support for object systems and can reason about their rewriting logic specifications are clearly of interest. Two candidate formalisms of this kind have been proposed. One is a version of the modal μ-calculus proposed by Lechner for reasoning about object-oriented Maude specifications [194,195,198], and another is Denker's object-oriented distributed temporal logic [90]. A direction recently explored by Ölveczky and supported by the model-checking features of the Real-Time Maude tool [267] is a timed linear time temporal logic suitable for reasoning about rewriting logic specifications of real-time systems [262]; in a similar vein, Beffara et al. have used rewrite rules and ELAN strategies to verify properties of timed automata [14]. An even more recent direction actively pursued at SRI is the development of an explicit state model checker to check linear temporal logic formulas on the general class of rewriting logic specifications executable in Maude; this model checker will be part of the upcoming Maude 2.0 distribution.

4 Semantic Framework Applications

The computational and logical interpretations of rewriting logic lead to applications that use it: as a *semantic framework*, in which different languages and models of computation are expressed; or as a *logical framework*, in which different logics and inference systems are likewise expressed [208]. We first discuss semantic framework applications.

4.1 Models of Computation

This section presents concrete evidence (in highly condensed form; see [223,227] for much more detailed discussions) for the thesis that a wide variety of models of computation, including concurrent ones, can be naturally and directly expressed as rewrite theories in rewriting logic. As a consequence, models hitherto quite different from each other can be naturally unified and interrelated within a common framework.

The following models of computation have been naturally expressed in rewriting logic: (1) equational programming, which is the special case of rewrite theories whose set of rules is empty and whose equations are Church-Rosser, possibly modulo some axioms A; (2) lambda calculi and combinatory reduction systems [218,192,193,295,292]; (3) labeled transition systems [218]; (4) grammars and string-rewriting systems [218]; (5) Petri nets, including place/transition nets, contextual nets, algebraic nets, colored nets, and timed Petri nets [218,223,293,297,268,289]; (6) Gamma and the Chemical Abstract Machine [218]; (7) CCS and LOTOS [230,208,314,45,89,311,309,201]; (8) the π calculus [316,292]; (9) concurrent objects and actors [218,220,300,302,304]; (10) the UNITY language [218]; (11) concurrent graph rewriting [223]; (12) dataflow [223]; (13) neural networks [223]; (14) real-time systems, including timed automata, timed transition systems, hybrid automata, and timed Petri nets [268,262]; and (15) the tile logic [146,147,135] model of synchronized concurrent computation [232,39,34,148].

Since the above specifications of models of computation as rewrite theories are typically executable, this suggests that rewriting logic is a very flexible *operational* semantic framework to specify the semantics of such models. What is not immediately apparent from the above list is that it is also a flexible *mathematical* semantic framework at the level of concurrency models. That is, quite often a well-known mathematical model of concurrency happens to be isomorphic to the initial model $\mathcal{T}_{\mathcal{R}}$ of the rewrite theory $\mathcal{R}$ axiomatizing that particular model, or at least closely related to such an initial model. Some examples will illustrate this point: (1) in [193] it is shown that for rewrite theories of the form $\mathcal{R} = (\Sigma, \emptyset, L, R)$, with the rules R left-linear, $\mathcal{T}_{\mathcal{R}}$ is isomorphic to a model based on residuals and permutation equivalence proposed by Boudol; (2) the same paper also shows that for $\mathcal{R} = (\Sigma, E, L, R)$ a rewrite theory axiomatizing an orthogonal combinatory reduction system, including the λ-calculus, a quotient of $\mathcal{T}_{\mathcal{R}}$ by a few additional equations is isomorphic to a well-known model of parallel reductions based on residuals and permutation equivalence; (3) the paper [297] shows in detail that for $\mathcal{R} = (\Sigma, E, L, R)$ a rewrite theory axiomatizing a place/transition net, $\mathcal{T}_{\mathcal{R}}$ is naturally isomorphic (in the categorical sense) to the Best-Devillers net process model—a result essentially known from the coincidence of $\mathcal{T}_{\mathcal{R}}$ with the Meseguer-Montanari algebraic model of nets [218] and the Degano-Meseguer-Montanari algebraic characterization of net processes—and then generalizes this natural isomorphism to one between $\mathcal{T}_{\mathcal{R}}$ and a Best-Devillers-like model for $\mathcal{R}$ the axiomatization of an algebraic net; (4) the papers [45,89] show that for $\mathcal{R} = (\Sigma, E, L, R)$ a rewrite theory axiomatizing CCS, a truly concurrent semantics causal model based on proved transition systems is isomorphic to a quotient of $\mathcal{T}_{\mathcal{R}}$ by a few additional axioms; (5) the paper [237] shows that for $\mathcal{R} = (\Sigma, E, L, R)$ a rewrite theory axiomatizing a concurrent object-oriented system satisfying reasonable requirements, a subcategory of $\mathcal{T}_{\mathcal{R}}$ is isomorphic to a partial order of events model which, for asynchronous object systems corresponding to actors, coincides with the finitary part of the Hewitt-Baker partial order of events model.

An important additional development in this area is the ρ-calculus of Cirstea and Kirchner [57,54,59,60]. This is a very general rewrite theory that can play for rewriting logic specifications a role similar to that played by the λ-calculus in functional computing; its generality is shown by the fact that ρ-terms generalize the rewriting logic proof terms defined in [218]. Furthermore, the ρ-calculus can simulate the λ-calculus itself. In fact, by replacing and generalizing the λ-calculus idea of function application by that of *rule application*, the ρ-calculus unifies both the λ-calculus and first-order rewriting. In analogy with λ-calculi, there are typed versions, including a simply typed ρ-calculus and a "ρ cube" [58,62].

4.2 Semantics of Programming Languages

Rewriting logic is a promising semantic framework for formally specifying programming languages as rewrite theories. Since those specifications usually can be executed in a rewriting logic language, they in fact become *interpreters* for the languages in question. In addition, such formal specifications allow both formal reasoning and a variety of formal analyses for the languages so specified.

The use of rewrite rules to define the semantics of programming languages is of course not new. In a higher-order version it goes back to the use of semantic equations in denotational semantics; in a first-order version, the power of equational specifications to give semantic definitions of conventional languages has been understood and used for a long time. However, both the lambda calculus and executable equational specifications implicitly assume that such language definitions can be given in terms of *functions*, and rely on the Church-Rosser property to reach the result of an execution. For sequential languages, by making the state of the computation explicit, a functional description of this kind can always be achieved. The situation becomes more difficult for languages that support highly concurrent and nondeterministic applications, and where the possibly nonterminating *interactions* between processes or components—as opposed to the computation of an output value from given inputs—are often the whole point of a program. Such languages and applications do not have a natural equational description in terms of functions, but do have a very natural rewriting logic semantics, not only operationally (by means of rewriting steps) but also denotationally ($\mathcal{T}_{\mathcal{R}}$ and related models).

Since structural operational semantics definitions can be used for languages not amenable to a functional description, it is natural to compare them with rewriting logic definitions. Their relationship has been discussed in detail in [208]. In fact, both "big-step" and "small-step" structural operational semantics definitions can be naturally regarded as special formats of corresponding rewrite theory definitions [208]. Tile models provide yet another systematic way of understanding structural operational semantics definitions as tile rewrite theories [146–148], which can then be mapped into rewriting logic for execution purposes [232,39,34]. There is also a close connection between rewriting logic and Mosses's modular structural operational semantics (MSOS) which has been recognized from the beginning [247,248], and that has led to ongoing work on a Maude Action Tool to execute MSOS definitions and Action Semantics definitions [32].

A number of encouraging case studies giving rewriting logic definitions of programming languages have already been carried out by different authors. Firstly, some of the models of computation discussed in Section 4.1 are so closely connected with languages that their rewriting logic specifications are also language specifications. Good examples are rewriting logic definitions of the lambda calculus and (mini-) ML, CCS, the π-calculus, and sketches of UNITY and Gamma, which are given in some of the references cited in Section 4.1. Secondly, the usefulness of rewriting strategies to specify program evaluations has been clearly demonstrated in ELAN specifications, for example of Prolog and of the functional-logic programming language Babel [320], and also in the Maude executable specifications for CCS developed by Bruni and Clavel [63,34], and by Verdejo and Martí-Oliet [311,310]. Thirdly, the fact that rewriting logic naturally supports concurrent objects has proved very useful in formally specifying a number of novel concurrent and mobile languages. For example, Ishikawa et al. have given a Maude specification of a representative subset of GAEA, a reflective concurrent logic programming language developed at ETL, Japan [164,163]. Mason and Talcott have used rewriting logic to give semantic definitions of actor languages, and to "compile away" certain language features by defining semantics-preserving translations between actor languages that are formalized as translations between their corresponding rewrite theories [212]. Van Baalen, Caldwell, and Mishra have used Maude to give a formal semantics to the DaAgent mobile agent system and to analyze key fault-tolerant protocols in that language [9]; their analysis has revealed mistakes and inconsistencies in the protocols' informal specifications. Yet another example is the formal executable specification in Maude of UPenn's PLAN active network programming language [234,322]. Maude itself has been used to define the semantics of its Mobile Maude extension [120]. Finally, Maude has been used not only to specify programming languages, but also to specify and verify microprocessors in work by Harman [154,155].

4.3 Distributed Architectures and Components

It is very important to detect errors and inconsistencies as early as possible in the software design cycle. For this reason, formal approaches that can increase the analytic power of architectural notations such as architectural description languages (ADLs) and object-oriented design formalisms like UML are quite valuable. A related concern is the formal specification and analysis of *distributed component architectures.*

Rewriting logic has been used by several authors in these areas to allow formal analysis of software designs and, in some cases, to support code generation from the associated executable specifications. Relevant work in this direction includes: (1) work of Nodelman and Talcott representing both the Wright architecture description language and its underlying CSP semantics in Maude; (2) work of Durán, Meseguer, and Talcott on semantic interoperation of heterogeneous software architectures based on their rewriting logic semantics [235] (see also Appendix E of [69]); (3) work of Wirsing and Knapp on the systematic transformation of UML diagrams and similar object-oriented notations into formal executable rewriting logic specifications in Maude, which can then be used to execute and formally analyze the designs, and even to generate code in a conventional language such as Java [326,185,186,327]; (4) work by Fernández and Toval formalizing in Maude the UML metamodel and its evolution [305,132], with applications to formal analysis and prototyping [131,306]; (5) work by Nakajima and Futatsugi on the transformation of GILO-2 scenario-based object-oriented design diagrams for execution and formal analysis [254]; (6) work by Talcott on a rewriting logic semantics for actor systems axiomatized by actor theories [300–304]; such systems can be extended by an algebra of *components*, that are encapsulated by interfaces, and that can include actors, messages, and other (sub-)components; in addition Talcott has developed methods to reason formally about such open component systems; (7) work by Denker, Meseguer, and Talcott on a general middleware architecture for composable distributed communication services such as fault-tolerance, security, and so on, that can be composed and can be dynamically added to selected subsets of a distributed communications system [96]; (8) work by Najm and Stefani giving a rewriting logic semantics to the operational subset of the Reference Model for Open Distributed Processing (RM-ODP) [249–251] (see also [128]); (9) work by Nakajima that uses rewriting logic specifications in CafeOBJ to formally specify the architecture of WEB-NMS, a Java/ORB implementation of a network management system [252]; and (10) work by Albarrán, Durán, and Vallecillo on interoperating Maude executable specifications with distributed component platforms such as CORBA and SOAP [1–3].

4.4 Specification and Analysis of Communication Protocols

Because of its flexibility to model distributed objects with different modes of communication and interaction, rewriting logic is very well suited to specify and analyze communication protocols, including cryptographic protocols, and, more generally, network software such as active network programming languages, active network algorithms, and network management systems.

Applications of this kind include: (1) work by researchers at Stanford, SRI, and at the Computer Communications Research Group at University of California Santa Cruz using Maude to analyze the early design of a new reliable broadcast protocol for active networks [91,92]; (2) work of Denker, Meseguer, and Talcott on the specification and analysis of cryptographic protocols using Maude [94,95] (see also [279]); (3) work of Basin and Denker on an experimental comparison of the advantages and disadvantages of using Maude versus using Haskell to analyze security protocols [13]; (4) work of Millen and Denker at SRI using Maude to give a formal semantics to their new cryptographic protocol specification language CAPSL, and to endow CAPSL with an execution and formal analysis environment [97–100]; (5) work of Wang, Gunter, and Meseguer using Maude to formally specify and analyze a PLAN active network algorithm [322]; (6) work by Ölvecky et al. using Real-Time Maude to specify and analyze the AER/NCA suite of active network protocol components for reliable multicast [263]; (7) work of Verdejo, Pita, and Martí-Oliet on the Maude specification and verification of the FireWire leader election protocol [312]; (8) work of Mason and Talcott on modeling, simulation and analysis of network architectures and communication protocols [213]; and (9) work of Pita and Martí-Oliet using the reflective features of Maude to specify some management processes of broadband telecommunication networks [273–275].

5 Logical Framework Applications

Rewriting logic is like a coin with two inseparable sides: one computational and another logical. The generality and expressiveness of rewriting logic as a semantic framework for concurrent computation has also a logical counterpart. Indeed, rewriting logic is also a promising *logical framework* in which many different logics and formal systems can be naturally represented and interrelated [208,209]. Using a rewriting logic implementation such representations can then be used to generate a wide range of formal tools.

5.1 Representing, Mapping, and Reasoning about Logics

The basic idea is that we can represent a logic $\mathcal{L}$ with a finitary syntax and inference system within rewriting logic by means of a representation map

$$\Phi : \mathcal{L} \longrightarrow \mathit{RWLogic}.$$

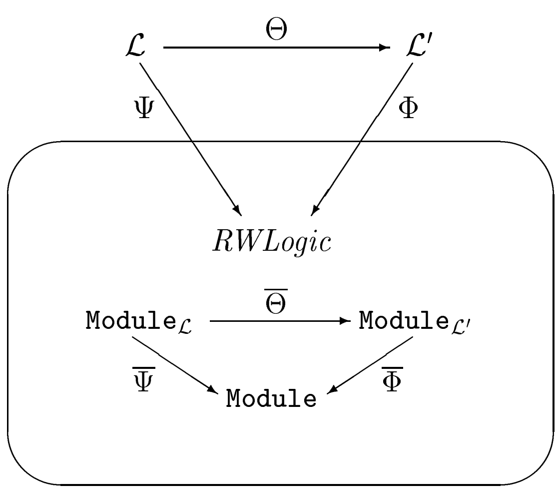
The map Φ should be *conservative*, that is, it should preserve and reflect theoremhood. The reason why rewriting logic is a good framework is that the formulas of a logic $\mathcal{L}$ can typically be equationally axiomatized by an equational theory, and the rules of inference can then be typically understood as rewrite rules, that may be conditional if the inference rules have "side conditions." Therefore, the mappings Φ are usually very simple and direct. Furthermore, using reflection we can define and execute a map Φ of this kind *inside rewriting logic itself* by means of an equationally defined map

$$\overline{\Phi} : \mathtt{Module}_{\mathcal{L}} \longrightarrow \mathtt{Module}.$$

The map $\overline{\Phi}$ can be defined by extending the universal theory $\mathcal{U}$, which has a sort `Module` representing rewrite theories (see Section 2), with the equational definition of a new sort $\mathtt{Module}_{\mathcal{L}}$ whose terms represent (finitely presentable) theories in the logic $\mathcal{L}$.

In fact, we can go a step further, and represent inside rewriting logic a mapping $\Theta : \mathcal{L} \longrightarrow \mathcal{L}'$ between any two finitary logics $\mathcal{L}$ and $\mathcal{L}'$ as an equationally defined function $\overline{\Theta} : \mathtt{Module}_{\mathcal{L}} \longrightarrow \mathtt{Module}_{\mathcal{L}'}$. If the map Θ is computable, then, by a metatheorem of Bergstra and Tucker it is possible to define the function $\overline{\Theta}$ by means of a finite set of Church-Rosser and terminating equations. That is, such functions can be effectively defined and executed within rewriting logic.

In summary, using reflection, mappings between logics, including maps representing other logics in rewriting logic, can be internalized and executed within rewriting logic, as indicated in the picture below.



There is yet another reason why rewriting logic is very useful for logical framework applications. Thanks to reflection and the existence of initial models, rewriting logic can not only be used as a logical framework in which the deduction of a logic $\mathcal{L}$ can be faithfully simulated, but also as a *metalogical framework* in which we can reason about the metalogical properties of a logic

$\mathcal{L}$. Basin, Clavel, and Meseguer have begun studying the use of reflection, induction, and Maude's inductive theorem prover enriched with reflective reasoning principles to prove such metalogical properties [10–12].

A good number of examples of representations of logics in rewriting logic have been given by different authors, often in the form of executable specifications, including: (1) the logics represented by Martí-Oliet and Meseguer in [208,209], including equational logic, Horn logic with equality, linear logic, logics with quantifiers, and any sequent calculus presentation of a logic for a very general notion of "sequent"; (2) the map $LinLogic \longrightarrow RWLogic$ in [208,209] representing propositional linear logic was subsequently specified in a reflective way in Maude by Clavel and Martí-Oliet [63,65]; (3) the map $HOL \longrightarrow Nuprl$ between the logics of the HOL and Nuprl theorem provers has been specified in Maude by Stehr, Naumov, and Meseguer [257,298]; (4) Dowek, Hardin, and Kirchner have presented (what obviously are) rewrite theories for doing deduction *modulo* an equational theory of equivalence between formulas specified by the equations E of the rewriting logic axiomatization, both for first-order and higher-order logics [109–111]; (5) the connections with rewriting logic of that work have been made explicit by Viry, who has given a coherent sequent calculus rewrite theory in this style in [317,318] (see also [101]); (6) Stehr and Meseguer have defined a natural representation map $PTS \longrightarrow RWLogic$ of pure type systems (a parametric family of higher-order logics generalizing the λ-cube) in rewriting logic [295]; and (7) Bruni, Meseguer, and Montanari have defined a mapping $Tile\ Logic \longrightarrow RWLogic$ from tile logic into rewriting logic that can be used to execute tile logic specifications [34,37–40].

5.2 Specifying and Building Formal Tools

Theorem provers and other formal tools have underlying *inference systems* that can be naturally specified and prototyped in rewriting logic. Furthermore, the strategy aspects of such tools and inference systems can then be specified by rewriting strategies. The researchers in the ELAN group have developed an impressive collection of rewriting logic specifications for different automated deduction inference systems, including the already-mentioned theorem proving modulo methods [109–111], logical languages, unification and narrowing [169,320], Knuth-Bendix completion with constraints [176], higher-order unification [15], combination of unification algorithms [277], constraint solving [46–50], and termination and tree-automata techniques [149,150]. In a somewhat similar vein, the work of Schorlemmer explores the relationships between rewriting logic and Levy and Agustí's general bi-rewriting approach to automated deduction [284–287].

In Maude, formal tools have typically a reflective design that, by metarepresenting theories as data, easily allows inference steps that may transform the object theory. Strategies are then rewrite theories controlling the application of such metalevel inference rules at the meta-metalevel. We have already mentioned in Section 3 several such tools that are part of the Maude formal environment, namely, an inductive theorem prover; Church-Rosser, coherence, and termination checkers, and a Knuth-Bendix completion tool [75–78,117,119,124]; plus the Real-Time Maude tool [267,262,269]. Also closely related to Maude itself is the Full Maude tool, which extends Maude with special syntax for object-oriented specifications, and with a rich *module algebra* of parameterized modules and module composition operations [121,116,127]. This method of building formal tools is not restricted to Maude-related tools: One can generate tools from their rewriting logic specifications for *any* finitary logic, such as: (1) a proof assistant built by Stehr for the Open Calculus of Constructions, which extends Coquand and Huet's calculus of constructions with equational reasoning and a flexible universe hierarchy [294]; (2) the Maude Action Tool [32] already mentioned in Section 4.2; (3) a CCS execution and verification environment developed by Verdejo and Martí-Oliet [311,310]; (4) a tool by Havelund and Roşu for testing linear temporal logic formulae on finite execution traces [157–160,280]; and (5) a tool by Fischer and Roşu to automatically check an abstract interpretation against user-given properties [137].

6 Language Implementations

Several language implementation efforts in France, Japan, and the US have adopted rewriting logic as their semantic basis and support executable rewriting logic specification and programming.

The ELAN language has been developed at LORIA (CNRS, INRIA, and Universities of Nancy) [169,320,25–27,19]. It has as modules *computational systems*, consisting of a rewrite theory and a *strategy* to guide the rewriting process [22,29,17,28]. As already discussed in Section 5, this group and their collaborators have developed a very impressive collection of examples and case studies in areas such as logic programming languages, constraint solving, higher-order unification, equational theorem-proving, and other such computational systems. Besides the ELAN interpreter, there is also a high-performance ELAN compiler, including compilation of AC-rewriting [243–246,179].

The CafeOBJ language implementation, developed at the Japan Advanced Institute of Science and Technology (JAIST) in Kanazawa [143,141,104,105,108], contains OBJ as its functional sublanguage, and supports object-oriented specifications. Furthermore, its semantics is multi-logical and includes hidden-sorted versions of equational and rewriting logic [102–105]. The CafeOBJ language has been the basis of an ambitious research effort—the Cafe Project—involving several research institutions in Japan, Europe and the US, as well as several Japanese industries, to promote formal methods applications in software engineering [138,142]. This project has achieved a distributable version of the language and further work on its semantics, a collection of specification libraries and case studies, an environment, and a collection of theorem proving tools supporting different forms of verification. Furthermore, a compiler has been developed in addition to the Cafe interpreter implementation [260,165].

The Maude language has been developed at SRI, in Menlo Park, California [220,80,69,74,71]. The equational logic underlying Maude's rewriting logic is membership equational logic [226,30,31], and gives rise to a sublanguage of *functional modules*. *System modules* specify general rewrite theories, and *object-oriented modules* provide syntactic sugar for object-oriented rewrite theories. These modules can be combined by module composition operations supported by Full Maude [116,127,122]. Maude's high-performance rewrite engine makes extensive use of advanced semicompilation techniques; there is also a high-performance experimental Maude compiler. In addition, Maude efficiently supports reflection through its `META-LEVEL` module [66,74]. Maude has been used in a wide range of applications, many of which have been discussed in this paper.

Acknowledgements

We would like to thank all our colleagues who have given us comments on earlier versions of this roadmap and have helped us in compiling the following bibliography; we thank specially Miguel Palomino, whose extensive help has made possible the existence of the bibliography.

References

[1] Antonio Albarrán, Francisco Durán, and Antonio Vallecillo. From Maude specifications to SOAP distributed implementations: A smooth transition. Manuscript, Universidad de Málaga, Spain, submitted for publication, 2001.

[2] Antonio Albarrán, Francisco Durán, and Antonio Vallecillo. Maude meets CORBA. In *Proceedings 2nd Argentine Symposium on Software Engineering*, Buenos Aires, Argentina, September 10–11, 2001.

[3] Antonio Albarrán, Francisco Durán, and Antonio Vallecillo. On the smooth implementation of component-based system specifications. In *Proceedings 6th ECOOP International Workshop on Component-Oriented Programming, WCOP'01*, Budapest, Hungary, June 2001.

[4] Nasreddine Aoumeur and Gunter Saake. On the specification and validation of cooperative information systems using an extended Maude. In Futatsugi et al. [140], pages 95–114.

[5] Egidio Astesiano, Manfred Broy, and Gianna Reggio. Algebraic specification of concurrent systems. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, IFIP State-of-the-Art Reports, pages 467–520. Springer-Verlag, 1999.

[6] Egidio Astesiano and Gianna Reggio. Algebraic specification of concurrency. In M. Bidoit and C. Choppy, editors, *Recent Trends in Data Type Specification, 8th Workshop on Specification of Abstract Data Types joint with the 3rd COMPASS Workshop, Dourdan, France, August 26–30, 1991, Selected Papers*, volume 655 of *Lecture Notes in Computer Science*, pages 1–39. Springer-Verlag, 1993.

[7] Egidio Astesiano and Gianna Reggio. Labelled transition logic: An outline. Technical Report DISI-TR-96-20, Dipartimento di Informatica e Scienze dell'Informazione, Università di Genoa, Italy, 1996.

[8] Egidio Astesiano and Gianna Reggio. On the relationship between labelled transition logic and rewriting logic. Technical Report DISI-TR-97-23, Dipartimento di Informatica e Scienze dell'Informazione, Università di Genoa, Italy, 1997.

[9] Jeffrey Van Baalen, James L. Caldwell, and Shivakant Mishra. Specifying and checking fault-tolerant agent-based protocols using Maude. In *First Goddard Workshop on Formal Approaches to Agent-Based Systems, Greenbelt, MD, USA, April 5–7, 2000, Proceedings*, Lecture Notes in Computer Science. To appear.

[10] David Basin, Manuel Clavel, and José Meseguer. Reflective metalogical frameworks. In *Proceedings of LFM'99: Workshop on Logical Frameworks and Meta-languages*, Paris, France, September 28, 1999. `http://www.cs.bell-labs.com/~felty/LFM99/`.

[11] David Basin, Manuel Clavel, and José Meseguer. Rewriting logic as a metalogical framework. In S. Kapoor and S. Prasad, editors, *Twentieth Conference on the Foundations of Software Technology and Theoretical Computer Science, New Delhi, India, December 13–15, 2000, Proceedings*, volume 1974 of *Lecture Notes in Computer Science*, pages 55–80. Springer-Verlag, 2000.

[12] David Basin, Manuel Clavel, and José Meseguer. Rewriting logic as a metalogical framework. Manuscript, Computer Science Laboratory, SRI International, submitted for publication, 2000.

[13] David Basin and Grit Denker. Maude versus Haskell: An experimental comparison in security protocol analysis. In Futatsugi [139], pages 235–256. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[14] Emmanuel Beffara, Olivier Bournez, Hassen Kacem, and Claude Kirchner. Verification of timed automata using rewrite rules and strategies. In N. Dershowitz and A. Frank, editors, *Proceedings BISFAI 2001, Seventh Biennial Bar-Ilan International Symposium on the Foundations of Artificial Intelligence*, Ramat-Gan, Israel, June 25–27, 2001.

[15] Peter Borovanský. Implementation of higher-order unification based on calculus of explicit substitutions. In M. Bartosek, J. Staudek, and J. Wiedermann, editors, *SOFSEM'95: Theory and Practice of Informatics, 22nd Seminar on Current Trends in Theory and Practice of Informatics, Milovy, Czech Republic, November 23 – December 1, 1995, Proceedings*, volume 1012 of *Lecture Notes in Computer Science*, pages 363–368. Springer-Verlag, 1995.

[16] Peter Borovanský. Controlling rewriting: Study and implementation of a strategy formalism. In Kirchner and Kirchner [167], pages 63–74. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[17] Peter Borovanský. *Le Contrôle de la Réécriture: Étude et Implantation d'un Formalisme de Stratégies.* PhD thesis, Université Henri Poincaré – Nancy I, October 1998.

[18] Peter Borovanský and Carlos Castro. Cooperation of constraint solvers: Using the new process control facilities of ELAN. In Kirchner and Kirchner [167], pages 379–398. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[19] Peter Borovanský, Horatiu Cirstea, Hubert Dubois, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, Christophe Ringeissen, and Marian Vittek. ELAN v 3.3 user manual, Third edition. Technical report, INRIA Lorraine & LORIA, Nancy, France, December 1998.

[20] Peter Borovanský, Salma Jamoussi, Pierre-Etienne Moreau, and Christophe Ringeissen. Handling ELAN rewrite programs via an exchange format. In Kirchner and Kirchner [167], pages 207–224. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[21] Peter Borovanský, Claude Kirchner, and Hélène Kirchner. Controlling rewriting by rewriting. In Meseguer [222], pages 168–188. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[22] Peter Borovanský, Claude Kirchner, and Hélène Kirchner. Strategies and rewriting in ELAN. In B. Gramlich and H. Kirchner, editors, *Proceedings of the CADE-14 Workshop on Strategies in Automated Deduction*, pages 13–24, Townsville, Australia, July 1997.

[23] Peter Borovanský, Claude Kirchner, and Hélène Kirchner. A functional view of rewriting and strategies for a semantics of ELAN. In M. Sato and Y. Toyama, editors, *The Third Fuji International Symposium on Functional and Logic Programming*, pages 143–167, Kyoto, Japan, April 1998. World Scientific.

[24] Peter Borovanský, Claude Kirchner, and Hélène Kirchner. Rewriting as a unified specification tool for logic and control: The ELAN language. In M. P. A.

Sellink, editor, *Second International Workshop on the Theory and Practice of Algebraic Specifications, Amsterdam, The Netherlands, September 25–26, 1997*, Electronic Workshops in Computing. Springer-Verlag, 1998. `http://www.ewic.org.uk/ewic/workshop/view.cfm/ASFSDF-97`.

[25] Peter Borovanský, Claude Kirchner, Hélène Kirchner, and Pierre-Etienne Moreau. ELAN from the rewriting logic point of view. *Theoretical Computer Science*, 2001. This volume.

[26] Peter Borovanský, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, and Christophe Ringeissen. An overview of ELAN. In Kirchner and Kirchner [167], pages 329–344. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[27] Peter Borovanský, Claude Kirchner, Hélène Kirchner, Pierre-Etienne Moreau, and Marian Vittek. ELAN: A logical framework based on computational systems. In Meseguer [222], pages 35–50. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[28] Peter Borovanský, Claude Kirchner, Hélène Kirchner, and Christophe Ringeissen. Rewriting with strategies in ELAN: A functional semantics. *International Journal of Foundations of Computer Science*, 2001. To appear.

[29] Peter Borovanský and Hélène Kirchner. Strategies of ELAN: Meta-interpretation and partial evaluation. In M. P. A. Sellink, editor, *Second International Workshop on the Theory and Practice of Algebraic Specifications, Amsterdam, The Netherlands, September 25–26, 1997*, Electronic Workshops in Computing. Springer-Verlag, 1998. `http://www.ewic.org.uk/ewic/workshop/view.cfm/ASFSDF-97`.

[30] Adel Bouhoula, Jean-Pierre Jouannaud, and José Meseguer. Specification and proof in membership equational logic. In M. Bidoit and M. Dauchet, editors, *TAPSOFT'97: Theory and Practice of Software Development, 7th International Joint Conference CAAP/FASE, Lille, France, April 14–18, 1997, Proceedings*, volume 1214 of *Lecture Notes in Computer Science*, pages 67–92. Springer-Verlag, 1997.

[31] Adel Bouhoula, Jean-Pierre Jouannaud, and José Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236:35–132, 2000.

[32] Christiano Braga, Hermann Haeusler, José Meseguer, and Peter Mosses. Maude Action Tool: Using reflection to map action semantics to rewriting logic. In T. Rus, editor, *Algebraic Methodology and Software Technology, 8th International Conference, AMAST 2000, Iowa City, Iowa, USA, May 20–27, 2000, Proceedings*, volume 1816 of *Lecture Notes in Computer Science*, pages 407–421. Springer-Verlag, 2000.

[33] Roberto Bruni. A logic for modular descriptions of asynchronous and synchronized concurrent systems. In Kirchner and Kirchner [167], pages 225–236. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[34] Roberto Bruni. *Tile Logic for Synchronized Rewriting of Concurrent Systems.* PhD thesis, Dipartimento di Informatica, Università di Pisa, 1999. Technical Report TD-1/99. `http://www.di.unipi.it/phd/tesi/tesi_1999/TD-1-99.ps.gz`.

[35] Roberto Bruni, David de Frutos-Escrig, Narciso Martí-Oliet, and Ugo Montanari. Bisimilarity congruences for open terms and term graphs via tile logic. In C. Palamidessi, editor, *CONCUR 2000, Concurrency Theory, 11th International Conference, University Park, PA, USA, August 22–25, 2000, Proceedings*, volume 1877 of *Lecture Notes in Computer Science*, pages 259–274. Springer-Verlag, 2000.

[36] Roberto Bruni, David de Frutos-Escrig, Narciso Martí-Oliet, and Ugo Montanari. Tile bisimilarity congruences for open terms and term graphs. Technical Report TR-00-06, Dipartimento di Informatica, Università di Pisa, 2000. `ftp://ftp.di.unipi.it/pub/techreports/TR-00-06.ps.Z`.

[37] Roberto Bruni, José Meseguer, and Ugo Montanari. Implementing tile systems: Some examples from process calculi. In P. Degano, U. Vaccaro, and G. Pirillo, editors, *Proceedings 6th Italian Conference on Theoretical Computer Science, ICTCS'98*, pages 168–179. World Scientific, 1998.

[38] Roberto Bruni, José Meseguer, and Ugo Montanari. Internal strategies in a rewriting implementation of tile systems. In Kirchner and Kirchner [167], pages 95–116. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[39] Roberto Bruni, José Meseguer, and Ugo Montanari. Process and term tile logic. Technical Report SRI-CSL-98-06, Computer Science Laboratory, SRI International, July 1998. Also TR-98-09, Dipartimento di Informatica, Università di Pisa. `ftp://ftp.di.unipi.it/pub/techreports/TR-98-09.ps.Z`.

[40] Roberto Bruni, José Meseguer, and Ugo Montanari. Executable tile specifications for process calculi. In J.-P. Finance, editor, *Fundamental Approaches to Software Engineering, Second International Conference, FASE'99, Held as Part of ETAPS'99, Amsterdam, The Netherlands, March 22–28, 1999, Proceedings*, volume 1577 of *Lecture Notes in Computer Science*, pages 60–76. Springer-Verlag, 1999.

[41] Roberto Bruni, José Meseguer, and Ugo Montanari. Symmetric monoidal and cartesian double categories as a semantic framework for tile logic. *Mathematical Structures in Computer Science*, 2000. To appear.

[42] Roberto Bruni and Ugo Montanari. Cartesian closed double categories, their lambda-notation, and the pi-calculus. In *Proceedings, Fourteenth Annual IEEE Symposium on Logic in Computer Science*, pages 246–265, Trento, July 2–5, 1999. IEEE Computer Society.

[43] Roberto Bruni, Ugo Montanari, and Francesca Rossi. An interactive semantics of logic programming. Manuscript, Dipartimento di Informatica, Università di Pisa, submitted for publication, 2000.

[44] Roberto Bruni, Ugo Montanari, and Vladimiro Sassone. Open ended systems, dynamic bisimulation and tile logic. In J. van Leeuwen, O. Watanabe, M. Hagiya, P. D. Mosses, and T. Ito, editors, *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics, International Conference IFIP TCS 2000 Sendai, Japan, August 17–19, 2000, Proceedings*, volume 1872 of *Lecture Notes in Computer Science*, pages 440–456. Springer-Verlag, 2000.

[45] Georgia Carabetta, Pierpaolo Degano, and Fabio Gadducci. CCS semantics via proved transition systems and rewriting logic. In Kirchner and Kirchner [167], pages 253–272. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[46] Carlos Castro. An approach to solving binary CSP using computational systems. In Meseguer [222], pages 245–264. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[47] Carlos Castro. Binary CSP solving as an inference process. In *Proceedings of the Eighth International Conference on Tools in Artificial Intelligence, ICTAI'96*, pages 462–463, Toulouse, France, November 1996.

[48] Carlos Castro. Constraint manipulation using rewrite rules and strategies. In *Proceedings of the Second ESSLLI Student Session, 9th European Summer School in Logic, Language and Information, ESSLLI'97*, pages 45–56, Aix-en-Provence, France, August 1997.

[49] Carlos Castro. Building constraint satisfaction problem solvers using rewrite rules and strategies. *Fundamenta Informaticae*, 34(3):263–293, September 1998.

[50] Carlos Castro. *Une Approche Déductive de la Résolution de Problèmes de Satisfaction de Contraintes*. PhD thesis, Université Henri Poincaré – Nancy I, 1998.

[51] María Victoria Cengarle. The rewriting logic institution. Technical Report 9801, Institut für Informatik, Ludwig-Maximilians-Universität München, May 1998.

[52] Anna Ciampolini, Evelina Lamma, Paola Mello, and Cesare Stefanelli. Distributed logic objects: A fragment of rewriting logic and its implementation. In Meseguer [222], pages 109–124. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[53] Horatiu Cirstea. Specifying authentication protocols in ELAN. In *Workshop on Modelling and Verification*, Besançon, France, December 1999.

[54] Horatiu Cirstea. *Calcul de Réécriture: Fondements et Applications*. PhD thesis, Université Henri Poincaré – Nancy I, October 2000.

[55] Horatiu Cirstea and Claude Kirchner. Theorem proving using computational systems: The case of the B predicate prover. Presented at *CCL'97 Workshop*, Schloss Dagstuhl, Germany, September 1997.

[56] Horatiu Cirstea and Claude Kirchner. Using rewriting and strategies for describing the B predicate prover. In B. Gramlich and F. Pfenning, editors, *Proceedings of the CADE-15 Workshop on Strategies in Automated Deduction*, Lindau, Germany, July 1998.

[57] Horatiu Cirstea and Claude Kirchner. Combining higher-order and first-order computations using ρ-calculus: Towards a semantics of ELAN. In D. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems 2*, pages 95–121. Research Studies Press/Wiley, 1999.

[58] Horatiu Cirstea and Claude Kirchner. The simply typed rewriting calculus. In Futatsugi [139], pages 23–41. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[59] Horatiu Cirstea and Claude Kirchner. The rewriting calculus — Part I. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):363–399, 2001.

[60] Horatiu Cirstea and Claude Kirchner. The rewriting calculus — Part II. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):401–434, 2001.

[61] Horatiu Cirstea, Claude Kirchner, and Luigi Liquori. Matching power. In A. Middeldorp, editor, *Rewriting Techniques and Applications, 12th International Conference, RTA 2001, Utrecht, The Netherlands, May 22-24, 2001, Proceedings*, volume 2051 of *Lecture Notes in Computer Science*, pages 77–92. Springer-Verlag, 2001.

[62] Horatiu Cirstea, Claude Kirchner, and Luigi Liquori. The rho cube. In F. Honsell and M. Miculan, editors, *Foundations of Software Science and Computation Structures, 4th International Conference, FOSSACS 2001, Held as Part of ETAPS 2001, Genova, Italy, April 2001, Proceedings*, volume 2030 of *Lecture Notes in Computer Science*, pages 168–183. Springer-Verlag, 2001.

[63] Manuel Clavel. *Reflection in General Logics and in Rewriting Logic, with Applications to the Maude Language*. PhD thesis, Universidad de Navarra, Spain, February 1998.

[64] Manuel Clavel. Reflection in general logics, rewriting logic, and Maude. In Kirchner and Kirchner [167], pages 317–328. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[65] Manuel Clavel. *Reflection in Rewriting Logic: Metalogical Foundations and Metaprogramming Applications*. CSLI Publications, 2000.

[66] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, and José Meseguer. Metalevel computation in Maude. In Kirchner and Kirchner [167], pages 3–24. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[67] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Language prototyping in the Maude metalanguage. Manuscript, Computer Science Laboratory, SRI International, November 1998.

[68] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude as a metalanguage. In Kirchner and Kirchner [167], pages 237–250. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[69] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude: Specification and programming in rewriting logic. Manual distributed as documentation of the Maude system, Computer Science Laboratory, SRI International. `http://maude.csl.sri.com/manual`, January 1999.

[70] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. The Maude system. In P. Narendran and M. Rusinowitch, editors, *Rewriting Techniques and Applications, 10th International Conference, RTA'99, Trento, Italy, July 2–4, 1999, Proceedings*, volume 1631 of *Lecture Notes in Computer Science*, pages 240–243. Springer-Verlag, 1999.

[71] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. A Maude tutorial. Tutorial distributed as documentation of the Maude system, Computer Science Laboratory, SRI International. Presented at the *European Joint Conference on Theory and Practice of Software, ETAPS 2000*, Berlin, Germany, March 25, 2000. `http://maude.csl.sri.com/tutorial`, March 2000.

[72] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Towards Maude 2.0. In Futatsugi [139], pages 297–318. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[73] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Using Maude. In T. Maibaum, editor, *Fundamental Approaches to Software Engineering, Third International Conference, FASE 2000, Held as Part of ETAPS 2000, Berlin, Germany, March/April 2000, Proceedings*, volume 1783 of *Lecture Notes in Computer Science*, pages 371–374. Springer-Verlag, 2000.

[74] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 2001. This volume.

[75] Manuel Clavel, Francisco Durán, Steven Eker, and José Meseguer. Building equational proving tools by reflection in rewriting logic. In Futatsugi [138]. `http://maude.csl.sri.com/papers`.

[76] Manuel Clavel, Francisco Durán, Steven Eker, and José Meseguer. Design and implementation of the Cafe prover and the Church-Rosser checker tools. Technical report, Computer Science Laboratory, SRI International, March 1998. `http://maude.csl.sri.com/papers`.

[77] Manuel Clavel, Francisco Durán, Steven Eker, and José Meseguer. Building equational proving tools by reflection in rewriting logic. In Futatsugi et al. [142], pages 1–31. `http://maude.csl.sri.com/papers`.

[78] Manuel Clavel, Francisco Durán, Steven Eker, José Meseguer, and Mark-Oliver Stehr. Maude as a formal meta-tool. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 — Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, September 20–24, 1999 Proceedings, Volume II*, volume 1709 of *Lecture Notes in Computer Science*, pages 1684–1703. Springer-Verlag, 1999.

[79] Manuel Clavel, Francisco Durán, and Narciso Martí-Oliet. Polytypic programming in Maude. In Futatsugi [139], pages 339–360. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[80] Manuel Clavel, Steven Eker, Patrick Lincoln, and José Meseguer. Principles of Maude. In Meseguer [222], pages 65–89. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[81] Manuel Clavel and José Meseguer. Axiomatizing reflective logics and languages. In G. Kiczales, editor, *Proceedings of Reflection'96, San Francisco, California, April 1996*, pages 263–288, 1996.

[82] Manuel Clavel and José Meseguer. Reflection and strategies in rewriting logic. In Meseguer [222], pages 125–147. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[83] Manuel Clavel and José Meseguer. Internal strategies in a reflective logic. In B. Gramlich and H. Kirchner, editors, *Proceedings of the CADE-14 Workshop on Strategies in Automated Deduction*, pages 1–12, Townsville, Australia, July 1997.

[84] Manuel Clavel and José Meseguer. Reflection in rewriting logic and its applications in the Maude language. In *Proceedings IMSA'97*, pages 128–139. Information-Technology Promotion Agency, Japan, 1997.

[85] Manuel Clavel and José Meseguer. Reflection in conditional rewriting logic. *Theoretical Computer Science*, 2001. This volume.

[86] Andrea Corradini and Fabio Gadducci. CPO models for infinite term rewriting. In V. S. Alagar and M. Nivat, editors, *Algebraic Methodology and Software Technology, 4th International Conference, AMAST'95, Montreal, Canada, July 3–7, 1995, Proceedings*, volume 936 of *Lecture Notes in Computer Science*, pages 368–384. Springer-Verlag, 1995.

[87] Andrea Corradini, Fabio Gadducci, and Ugo Montanari. Relating two categorical models of term rewriting. In J. Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA'95, Kaiserslautern, Germany, April 5–7, 1995, Proceedings*, volume 914 of *Lecture Notes in Computer Science*, pages 225–240. Springer-Verlag, 1995.

[88] Andrea Corradini, Reiko Heckel, and Ugo Montanari. Tile transition systems as structured coalgebras. In G. Ciobanu and G. Paun, editors, *Fundamentals of Computation Theory, 12th International Symposium, FCT'99, Iasi, Romania, August 30 – September 3, 1999, Proceedings*, volume 1684 of *Lecture Notes in Computer Science*, pages 13–38. Springer-Verlag, 1999.

[89] Pierpaolo Degano, Fabio Gadducci, and Corrado Priami. A causal semantics for CCS via rewriting logic. Manuscript, Dipartimento di Informatica, Università di Pisa, submitted for publication, 2000.

[90] Grit Denker. From rewrite theories to temporal logic theories. In Kirchner and Kirchner [167], pages 273–294. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[91] Grit Denker, José J. García-Luna-Aceves, José Meseguer, Peter Csaba Ölveczky, Jyoti Raju, Brad Smith, and Carolyn L. Talcott. Specification and analysis of a reliable broadcasting protocol in Maude. In B. Hajek and R. S. Sreenivas, editors, *Proceedings 37th Allerton Conference on Communication, Control and Computation*, pages 738–747. University of Illinois, 1999. `http://maude.csl.sri.com/casestudies/rbp`.

[92] Grit Denker, José J. García-Luna-Aceves, José Meseguer, Peter Csaba Ölveczky, Jyoti Raju, Brad Smith, and Carolyn L. Talcott. Specifying a reliable broadcasting protocol in Maude. Technical report, Computer Science Laboratory, SRI International, 1999. `http://www.csl.sri.com/casestudies/rbp`.

[93] Grit Denker and Martin Gogolla. Translating TROLL *light* concepts to Maude. In H. Ehrig and F. Orejas, editors, *Recent Trends in Data Type Specification, 9th Workshop on Specification of Abstract Data Types Joint with the 4th COMPASS Workshop, Caldes de Malavella, Spain, October 1992, Selected Papers*, volume 785 of *Lecture Notes in Computer Science*, pages 173–187. Springer-Verlag, 1994.

[94] Grit Denker, José Meseguer, and Carolyn L. Talcott. Protocol specification and analysis in Maude. In N. Heintze and J. Wing, editors, *Proceedings of Workshop on Formal Methods and Security Protocols, June 25, 1998, Indianapolis, Indiana*, 1998. `http://www.cs.bell-labs.com/who/nch/fmsp/index.html`.

[95] Grit Denker, José Meseguer, and Carolyn L. Talcott. Formal specification and analysis of active networks and communication protocols: The Maude experience. In D. Maughan, G. Koob, and S. Saydjari, editors, *Proceedings DARPA Information Survivability Conference and Exposition, DISCEX 2000, Hilton Head Island, South Carolina, January 25–27, 2000*, pages 251–265. IEEE Computer Society Press, 2000. `http://schafercorp-ballston.com/discex/`.

[96] Grit Denker, José Meseguer, and Carolyn L. Talcott. Rewriting semantics of meta-objects and composable distributed services. In Futatsugi [139], pages 407–427. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[97] Grit Denker and Jon Millen. CAPSL and CIL language design: A common authentication protocol specification language and its intermediate language. Technical Report SRI-CSL-99-02, Computer Science Laboratory, SRI International, 1999. `http://www.csl.sri.com/~denker/pub_99.html`.

[98] Grit Denker and Jon Millen. CAPSL intermediate language. In N. Heintze and E. Clarke, editors, *Proceedings of Workshop on Formal Methods and Security Protocols, FMSP'99, July 1999, Trento, Italy*, 1999. `http://www.cs.bell-labs.com/who/nch/fmsp99/program.html`.

[99] Grit Denker and Jon Millen. CAPSL integrated protocol environment. In D. Maughan, G. Koob, and S. Saydjari, editors, *Proceedings DARPA Information Survivability Conference and Exposition, DISCEX 2000, Hilton Head Island, South Carolina, January 25-27, 2000*, pages 207–222. IEEE Computer Society Press, 2000. `http://schafercorp-ballston.com/discex/`.

[100] Grit Denker and Jon Millen. The CAPSL integrated protocol environment. Technical Report SRI-CSL-2000-02, Computer Science Laboratory, SRI International, 2000. `http://www.csl.sri.com/~denker/pub_99.html`.

[101] Eric Deplagne. Sequent calculus viewed modulo. In C. Pilière, editor, *Proceedings of the Fifth ESSLLI Student Session*, pages 66–76. University of Birmingham, August 2000. `http://www.loria.fr/publications/2000/A00-R-256/A00-R-256.ps`.

[102] Răzvan Diaconescu. Behavioural rewriting logic: Semantic foundations and proof theory. Manuscript, submitted for publication. `http://ldl-www.jaist.ac.jp/cafeobj/documents.html`, October 1996.

[103] Răzvan Diaconescu. Foundations of behavioural specification in rewriting logic. In Meseguer [222], pages 225–244. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[104] Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report. The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification*, volume 6 of *AMAST Series in Computing*. World Scientific, 1998.

[105] Răzvan Diaconescu and Kokichi Futatsugi. Logical foundations of CafeOBJ. *Theoretical Computer Science*, 2001. This volume.

[106] Răzvan Diaconescu, Kokichi Futatsugi, and Shusaku Iida. Component-based algebraic specification and verification in CafeOBJ. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 — Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, September 20–24, 1999 Proceedings, Volume II*, volume 1709 of *Lecture Notes in Computer Science*, pages 1644–1663. Springer-Verlag, 1999.

[107] Răzvan Diaconescu, Kokichi Futatsugi, and Shusaku Iida. CafeOBJ jewels. In Futatsugi et al. [142], pages 33–60.

[108] Răzvan Diaconescu, Kokichi Futatsugi, Makoto Ishisone, Toshimi Sawada, and Ataru T. Nakagawa. An overview of CafeOBJ. In Kirchner and Kirchner [167], pages 75–88. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[109] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. Technical Report RR-3400, INRIA, Institut National de Recherche en Informatique et en Automatique, April 1998. `ftp://ftp.inria.fr/INRIA/publication/RR/RR-3400.ps.gz`.

[110] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher order unification via explicit substitutions. *Information and Computation*, 157(1/2):183–235, 2000.

[111] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. HOL-$\lambda\sigma$: An intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, 11(1):21–45, 2001.

[112] Hubert Dubois and Hélène Kirchner. Actions and plans in ELAN. In B. Gramlich and F. Pfenning, editors, *Proceedings of the CADE-15 Workshop on Strategies in Automated Deduction*, pages 35–45, Lindau, Germany, July 1998. Also Technical Report LORIA 98-R-275.

[113] Hubert Dubois and Hélène Kirchner. Modelling planning problems with rules and strategies. Technical Report 99-R-029, LORIA, Nancy, France, March 1999. Poster Session at JFPLC'99, Lyon, France, June 1999.

[114] Hubert Dubois and Hélène Kirchner. Objects, rules, and strategies in ELAN. In *Proceedings of the Second AMAST Workshop on Algebraic Methods in Language Processing*, Iowa City, Iowa, USA, 1999.

[115] Hubert Dubois and Hélène Kirchner. Rule based programming with constraints and strategies. In K. R. Apt, A. Kakas, E. Monfroy, and F. Rossi, editors, *New Trends in Constraints, Joint ERCIM/Compulog Net Workshop, Paphos, Cyprus, October 25-27, 1999, Selected Papers*, volume 1865 of *Lecture Notes in Artificial Intelligence*, pages 274–297. Springer-Verlag, 2000.

[116] Francisco Durán. *A Reflective Module Algebra with Applications to the Maude Language*. PhD thesis, Universidad de Málaga, Spain, June 1999. `http://maude.csl.sri.com/papers`.

[117] Francisco Durán. Coherence checker and completion tools for Maude specifications. Manuscript, Computer Science Laboratory, SRI International, `http://maude.csl.sri.com/papers`, 2000.

[118] Francisco Durán. The extensibility of Maude's module algebra. In T. Rus, editor, *Algebraic Methodology and Software Technology, 8th International Conference, AMAST 2000, Iowa City, Iowa, USA, May 20–27, 2000, Proceedings*, volume 1816 of *Lecture Notes in Computer Science*, pages 422–437. Springer-Verlag, 2000.

[119] Francisco Durán. Termination checker and Knuth-Bendix completion tools for Maude equational specifications. Manuscript, Computer Science Laboratory, SRI International, `http://maude.csl.sri.com/papers`, 2000.

[120] Francisco Durán, Steven Eker, Patrick Lincoln, and José Meseguer. Principles of Mobile Maude. In D. Kotz and F. Mattern, editors, *Agent Systems, Mobile Agents, and Applications, Second International Symposium on Agent Systems and Applications and Fourth International Symposium on Mobile Agents, ASA/MA 2000, Zurich, Switzerland, September 13–15, 2000, Proceedings*, volume 1882 of *Lecture Notes in Computer Science*, pages 73–85. Springer-Verlag, 2000.

[121] Francisco Durán and José Meseguer. An extensible module algebra for Maude. In Kirchner and Kirchner [167], pages 185–206. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[122] Francisco Durán and José Meseguer. The Maude specification of Full Maude. Manuscript, Computer Science Laboratory, SRI International, `http://maude.csl.sri.com/papers`, February 1999.

[123] Francisco Durán and José Meseguer. Structured theories and institutions. In M. Hofmann, G. Rosolini, and D. Pavlović, editors, *Proceedings of 8th Conference on Category Theory and Computer Science, Edinburgh, Scotland, September 1999*, volume 29 of *Electronic Notes in Theoretical Computer Science*, pages 71–90. Elsevier, 1999. `http://www.elsevier.nl/locate/entcs/volume29.html`.

[124] Francisco Durán and José Meseguer. A Church-Rosser checker tool for Maude equational specifications. Manuscript, Computer Science Laboratory, SRI International, `http://maude.csl.sri.com/papers`, 2000.

[125] Francisco Durán and José Meseguer. Parameterized theories and views in Full Maude 2.0. In Futatsugi [139], pages 319–337. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[126] Francisco Durán and José Meseguer. Structured theories and institutions. Manuscript, Computer Science Laboratory, SRI International, submitted for publication, 2000.

[127] Francisco Durán and José Meseguer. Maude's module algebra. *Theoretical Computer Science*, 2001. To appear. `http://maude.csl.sri.com/papers`.

[128] Francisco Durán and Antonio Vallecillo. Writing ODP enterprise specifications in Maude. In *Proceedings Workshop On Open Distributed Processing: Enterprise, Computation, Knowledge, Engineering and Realisation, WOODPECKER 2001*, Setúbal, Portugal, July 2001. `http://www.lcc.uma.es/~av/Publicaciones/01/ITI-2001-8.pdf`.

[129] Steven Eker. Fast matching in combination of regular equational theories. In Meseguer [222], pages 90–108. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[130] Steven Eker. Term rewriting with operator evaluation strategy. In Kirchner and Kirchner [167], pages 45–62. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[131] José Luis Fernández and Ambrosio Toval. Can intuition become rigorous? Foundations for UML model verification tools. In F. M. Titsworth, editor, *International Symposium on Software Reliability Engineering*, pages 344–355, San Jose, California, October 8–11, 2000. IEEE Press.

[132] José Luis Fernández and Ambrosio Toval. Seamless formalizing the UML semantics through metamodels. In K. Siau and T. Halpin, editors, *Unified Modeling Language: Systems Analysis, Design, and Development Issues*, pages 224–248. Idea Group Publishing, 2001.

[133] Gianluigi Ferrari and Ugo Montanari. A tile-based coordination view of asynchronous pi-calculus. In I. Privara and P. Ruzicka, editors, *Mathematical Foundations of Computer Science 1997, 22nd International Symposium, MFCS'97, Bratislava, Slovakia, August 25–29, 1997, Proceedings*, volume 1295 of *Lecture Notes in Computer Science*, pages 52–70. Springer-Verlag, 1997.

[134] Gianluigi Ferrari and Ugo Montanari. Tiles for concurrent and located calculi. In C. Palamidessi and J. Parrow, editors, *Proceedings EXPRESS'97*, volume 7 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1997. `http://www.elsevier.nl/locate/entcs/volume7.html`.

[135] Gianluigi Ferrari and Ugo Montanari. Tile formats for located and mobile systems. *Information and Computation*, 156(1/2):173–235, 2000.

[136] José Luiz Fiadeiro, Tom Maibaum, Narciso Martí-Oliet, José Meseguer, and Isabel Pita. Towards a verification logic for rewriting logic. In D. Bert, C. Choppy, and P. Mosses, editors, *Recent Trends in Algebraic Development Techniques, 14th International Workshop, WADT'99, Chateau de Bonas, France, September 15–18, 1999, Selected Papers*, volume 1827 of *Lecture Notes in Computer Science*, pages 438–458. Springer-Verlag, 2000.

[137] Bernd Fischer and Grigore Roşu. Interpreting abstract interpretations in membership equational logic. Technical Report RIACS 01.16, Research Institute for Advanced Computer Science, May 2001.

[138] Kokichi Futatsugi, editor. *Proceedings of the CafeOBJ Symposium '98, Numazu, Japan*. CafeOBJ Project, April 1998.

[139] Kokichi Futatsugi, editor. *Proceedings Third International Workshop on Rewriting Logic and its Applications, WRLA 2000, Kanazawa, Japan, September 18–20, 2000*, volume 36 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2000. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[140] Kokichi Futatsugi, Joseph A. Goguen, and José Meseguer, editors. *OBJ/CafeOBJ/Maude Workshop at Formal Methods '99: Formal Specification, Proof, and Applications*. Theta, 1999.

[141] Kokichi Futatsugi and Ataru T. Nakagawa. An overview of CAFE specification environment — An algebraic approach for creating, verifying, and maintaining formal specifications over networks. In *First International Conference on*

Formal Engineering Methods, Hiroshima, Japan, November 1997. http://ldl-www.jaist.ac.jp/cafeobj/documents.html.

[142] Kokichi Futatsugi, Ataru T. Nakagawa, and Tetsuo Tamai, editors. *Cafe: An Industrial-Strength Algebraic Formal Method*. Elsevier, 2000.

[143] Kokichi Futatsugi and Toshimi Sawada. Cafe as an extensible specification environment. In *Proceedings of the Kunming International CASE Symposium, Kunming, China*, November 1994.

[144] Fabio Gadducci. *On the Algebraic Approach to Concurrent Term Rewriting*. PhD thesis, Dipartimento di Informatica, Università di Pisa, March 1996. Technical Report TD-2/96.

[145] Fabio Gadducci and Ugo Montanari. Enriched categories as models of computation. In A. De Santis, editor, *Proceedings 5th Italian Conference on Theoretical Computer Science, Ravello*, pages 20–42. World Scientific, 1995.

[146] Fabio Gadducci and Ugo Montanari. Tiles, rewriting rules, and CCS. In Meseguer [222], pages 1–19. http://www.elsevier.nl/locate/entcs/volume4.html.

[147] Fabio Gadducci and Ugo Montanari. The tile model. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*. The MIT Press, 2000. http://www.di.unipi.it/~ugo/festschrift.ps.

[148] Fabio Gadducci and Ugo Montanari. Comparing logics for rewriting: Rewriting logic, action calculi and tile logic. *Theoretical Computer Science*, 2001. This volume.

[149] Thomas Genet. *Contraintes d'Ordre et Automates d'Arbres pour les Preuves de Terminaison*. PhD thesis, Université Henri Poincaré – Nancy I, 1998.

[150] Thomas Genet. Decidable approximations of sets of descendants and sets of normal forms. In T. Nipkow, editor, *Rewriting Techniques and Applications, 9th International Conference, RTA'98, Tsukuba, Japan, March 30–April 1, 1998, Proceedings*, volume 1379 of *Lecture Notes in Computer Science*, pages 151–165. Springer-Verlag, 1998.

[151] Fausto Giunchiglia. The OMRS project: State of the art and future developments. In Kirchner and Kirchner [167]. http://www.elsevier.nl/locate/entcs/volume15.html.

[152] Joseph A. Goguen, Kai Lin, and Grigore Roşu. Behavioral and coinductive rewriting. In Futatsugi [139], pages 1–22. http://www.elsevier.nl/locate/entcs/volume36.html.

[153] Juan Carlos González-Moreno, M. Teresa Hortalá-González, Francisco J. López-Fraguas, and Mario Rodríguez-Artalejo. An approach to declarative programming based on a rewriting logic. *Journal of Logic Programming*, 40:47–87, 1999.

[154] Neal A. Harman. Correctness and verification of hardware systems using Maude. Technical Report 3-2000, Department of Computer Science, University of Wales Swansea, 2000. `http://www-compsci.swan.ac.uk/reports/yr2000/CSR3-2000.pdf`.

[155] Neal A. Harman. Verifying a simple pipelined microprocessor using Maude. Technical Report 4-2000, Department of Computer Science, University of Wales Swansea, 2000. `http://www-compsci.swan.ac.uk/reports/yr2000/CSR4-2000.pdf`.

[156] Neal A. Harman. Verifying a microprocessor using Maude. In M. Cerioli, P. D. Mosses, and G. Reggio, editors, *Proceedings WADT/CoFI'01, 15th International Workshop on Algebraic Development Techniques and General Workshop of the CoFI WG*, Genova, Italy, April 1–3, 2001.

[157] Klaus Havelund and Grigore Roşu. Java PathExplorer — A runtime verification tool. In *Proceedings 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space, ISAIRAS'01*, Montreal, Canada, June 18–22, 2001.

[158] Klaus Havelund and Grigore Roşu. Monitoring Java programs with Java PathExplorer. In *Proceedings First Workshop on Runtime Verification, RV'01, Paris, France, July 23, 2001*, volume 55 (2) of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2001. `http://www.elsevier.nl/locate/entcs/volume55.html`.

[159] Klaus Havelund and Grigore Roşu. Monitoring programs using rewriting. Technical report, Research Institute for Advanced Computer Science, 2001.

[160] Klaus Havelund and Grigore Roşu. Testing linear temporal logic formulae on finite execution traces. Technical Report RIACS 01.08, Research Institute for Advanced Computer Science, May 2001.

[161] Hendrik Hilberdink. Foundations for rewriting logic. In Futatsugi [139], pages 43–69. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[162] Shusaku Iida, Michihiro Matsumoto, Răzvan Diaconescu, Kokichi Futatsugi, and Dorel Lucanu. Concurrent object composition in CafeOBJ. Technical Report JAIST IS-RR-98-0009S, Japan Advanced Institute of Science and Technology, 1998. `http://ldl-www.jaist.ac.jp/cafeobj/documents.html`.

[163] Hiroshi Ishikawa, Kokichi Futatsugi, and Takuo Watanabe. An operational semantics of GAEA in CafeOBJ. In Futatsugi et al. [140], pages 213–227.

[164] Hiroshi Ishikawa, José Meseguer, Takuo Watanabe, Kokichi Futatsugi, and Hideyuki Nakashima. On the semantics of GAEA — An object-oriented specification of a concurrent reflective language in rewriting logic. In *Proceedings IMSA'97*, pages 70–109. Information-Technology Promotion Agency, Japan, 1997.

[165] Makoto Ishisone and Toshimi Sawada. Brute: Brute force rewriting engine. In Futatsugi [138].

[166] Jean-Pierre Jouannaud. Membership equational logic, calculus of inductive constructions, and rewrite logic. In Kirchner and Kirchner [167], pages 89–94. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[167] Claude Kirchner and Hélène Kirchner, editors. *Proceedings Second International Workshop on Rewriting Logic and its Applications, WRLA'98, Pont-à-Mousson, France, September 1–4, 1998*, volume 15 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1998. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[168] Claude Kirchner, Hélène Kirchner, and Marian Vittek. Implementing computational systems with constraints. In P. Kanellakis, J.-L. Lassez, and V. Saraswat, editors, *Proceedings First Workshop on Principles and Practice of Constraint Programming*, pages 166–175, Brown University, Providence, RI, USA, 1993.

[169] Claude Kirchner, Hélène Kirchner, and Marian Vittek. Designing constraint logic programming languages using computational systems. In V. Saraswat and P. van Hentenryck, editors, *Principles and Practice of Constraint Programming: The Newport Papers*, pages 133–160. The MIT Press, 1995.

[170] Claude Kirchner and Christophe Ringeissen. Rule-based constraint programming. *Fundamenta Informaticae*, 34(3):225–262, September 1998.

[171] Hélène Kirchner. On the use of constraints in automated deduction. In A. Podelski, editor, *Constraint Programming: Basics and Trends, 1994 Chatillon Spring School, Chatillon-sur-Seine, France, May 16–20, 1994, Selected Papers*, volume 910 of *Lecture Notes in Computer Science*, pages 128–146. Springer-Verlag, 1995.

[172] Hélène Kirchner. Some extensions of rewriting. In H. Comon and J.-P. Jouannaud, editors, *Term Rewriting, French Spring School of Theoretical Computer Science, Font Romeaux, France, May 17–21, 1993, Advanced Course*, volume 909 of *Lecture Notes in Computer Science*, pages 54–73. Springer-Verlag, 1995.

[173] Hélène Kirchner. ELAN. In F. Fages, editor, *JFPLC'99, Journées Francophones de Programmation Logique et Programmation par Constraintes*, pages 241–248, Lyon, France, May 1999. HERMES Science Publications. Also Technical Report 99-R-129, LORIA, Nancy, France.

[174] Hélène Kirchner. Term rewriting. In E. Astesiano, H.-J. Kreowski, and B. Krieg-Brückner, editors, *Algebraic Foundations of Systems Specification*, IFIP State-of-the-Art Reports, pages 273–320. Springer-Verlag, 1999.

[175] Hélène Kirchner and Isabelle Gnaedig. Termination and normalisation under strategy — Proofs in ELAN. In Futatsugi [139], pages 93–115. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[176] Hélène Kirchner and Pierre-Etienne Moreau. Prototyping completion with constraints using computational systems. In J. Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA'95, Kaiserslautern, Germany, April 5–7, 1995, Proceedings*, volume 914 of *Lecture Notes in Computer Science*, pages 438–443. Springer-Verlag, 1995.

[177] Hélène Kirchner and Pierre-Etienne Moreau. A reflective extension of ELAN. In Meseguer [222], pages 148–167. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[178] Hélène Kirchner and Pierre-Etienne Moreau. Non-deterministic computations in ELAN. In J. L. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques, 13th International Workshop, WADT'98, Lisbon, Portugal, April 2–4, 1998, Selected Papers*, volume 1589 of *Lecture Notes in Computer Science*, pages 168–182. Springer-Verlag, 1999.

[179] Hélène Kirchner and Pierre-Etienne Moreau. Promoting rewriting to a programming language: A compiler for non-deterministic rewrite programs in associative-commutative theories. *Journal of Functional Programming*, 11(2):207–251, 2001.

[180] Hélène Kirchner and Christophe Ringeissen. Combining symbolic constraint solvers on algebraic domains. *Journal of Symbolic Computation*, 18(2):113–155, 1994.

[181] Hélène Kirchner and Christophe Ringeissen. Constraint solving by narrowing in combined algebraic domains. In P. van Hentenryck, editor, *Proceedings of 11th International Conference on Logic Programming*, pages 617–631, Santa Margherita Ligure, Italy, 1994. The MIT Press.

[182] Hélène Kirchner and Christophe Ringeissen. Executing CASL equational specifications with the ELAN rewrite engine. Technical Report 99-R-278, LORIA, Nancy, France, 1999.

[183] Hélène Kirchner and Laurent Vigneron. Deduction with constraints for theory reasoning: Completeness and simplification problems. In *Proceedings of CADE-12 Workshop: Theory Reasoning in Automated Deduction*, Nancy, France, 1994.

[184] Alexander Knapp. Case studies with CafeOBJ. In Futatsugi [138].

[185] Alexander Knapp. Generating rewrite theories from UML collaborations. In Futatsugi et al. [142], pages 97–120.

[186] Alexander Knapp. *A Formal Approach to Object-Oriented Software Engineering*. Shaker Verlag, Aachen, Germany, 2001. PhD thesis, Institut für Informatik, Universität München, 2000.

[187] Piotr Kosiuczenko and Martin Wirsing. Timed rewriting logic for the specification of time-sensitive systems. In H. Schwichtenberg, editor, *Logic of Computation, Proceedings of the NATO Advanced Study Institute on Logic of Computation, Held in Marktoberdorf, Germany, July 25 – August 6, 1997*,

volume 157 of *NATO ASI Series F: Computer and Systems Sciences*, pages 229–264. Springer-Verlag, 1997.

[188] Piotr Kosiuczenko and Martin Wirsing. Timed rewriting logic with an application to object-based specification. *Science of Computer Programming*, 28(2–3):225–246, April 1997.

[189] Marija Kulaš and Christoph Beirle. Defining Standard Prolog in rewriting logic. In Futatsugi [139], pages 155–171. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[190] Hironobu Kuruma and Kokichi Futatsugi. Incremental specification based on the combination of data types. In Futatsugi et al. [140], pages 95–114.

[191] Christopher Landauer. Discrete event systems in rewriting logic. In Meseguer [222], pages 309–320. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[192] Cosimo Laneve and Ugo Montanari. Axiomatizing permutation equivalence in the λ-calculus. In H. Kirchner and G. Levi, editors, *Algebraic and Logic Programming, Third International Conference, Volterra, Italy, September 2–4, 1992, Proceedings*, volume 632 of *Lecture Notes in Computer Science*, pages 350–363. Springer-Verlag, 1992.

[193] Cosimo Laneve and Ugo Montanari. Axiomatizing permutation equivalence. *Mathematical Structures in Computer Science*, 6(3):219–249, June 1996.

[194] Ulrike Lechner. Object-oriented specifications of distributed systems in the μ-calculus and Maude. In Meseguer [222], pages 384–403. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[195] Ulrike Lechner. *Object-Oriented Specification of Distributed Systems*. PhD thesis, Fakultät für Mathematik und Informatik, Universität Passau, June 1997.

[196] Ulrike Lechner. Constructs, concepts, and criteria for reuse in concurrent object-oriented languages. In E. Astesiano, editor, *Fundamental Approaches to Software Engineering, First International Conference, FASE'98, Held as Part of ETAPS'98, Lisbon, Portugal, March 28 – April 4, 1998, Proceedings*, volume 15 of *Lecture Notes in Computer Science*, pages 171–187. Springer-Verlag, 1998.

[197] Ulrike Lechner. Object-oriented specification of distributed systems. In Kirchner and Kirchner [167], pages 405–414. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[198] Ulrike Lechner and Christian Lengauer. Modal μ-Maude — Properties and specification of concurrent objects. In B. Freitag, C. B. Jones, C. Lengauer, and H.-J. Schek, editors, *Object Orientation with Parallelism and Persistence*, pages 41–62. Kluwer Academic Publishers, 1996.

[199] Ulrike Lechner, Christian Lengauer, Friederike Nickl, and Martin Wirsing. (Objects + concurrency) & reusability — A proposal to circumvent the inheritance anomaly. In P. Cointe, editor, *ECOOP'96 - Object-Oriented Programming, 10th European Conference, Linz, Austria, July 8–12, 1996, Proceedings*, volume 1098 of *Lecture Notes in Computer Science*, pages 232–247. Springer-Verlag, 1996.

[200] Ulrike Lechner, Christian Lengauer, and Martin Wirsing. An object-oriented airport. In E. Astesiano, G. Reggio, and A. Tarlecki, editors, *Recent Trends in Data Type Specification, 10th Workshop on Specification of Abstract Data Types Joint with the 5th COMPASS Workshop, S. Margherita, Italy, May 30 – June 3, 1994, Selected Papers*, volume 906 of *Lecture Notes in Computer Science*, pages 351–367. Springer-Verlag, 1995.

[201] Martin Leucker and Thomas Noll. Rewriting logic as a framework for generic verification tools. In Futatsugi [139], pages 117–133. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[202] Patrick Lincoln, Narciso Martí-Oliet, and José Meseguer. Specification, transformation, and programming of concurrent systems in rewriting logic. In G. E. Blelloch, K. M. Chandy, and S. Jagannathan, editors, *Specification of Parallel Algorithms, DIMACS Workshop, May 9–11, 1994*, volume 18 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 309–339. American Mathematical Society, 1994.

[203] Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Livio Ricciulli. Compiling rewriting onto SIMD and MIMD/SIMD machines. In C. Halatsis, D. Maritsas, G. Philokyprou, and S. Theodoridis, editors, *PARLE'94 Parallel Architectures and Languages Europe, 6th International PARLE Conference, Athens, Greece, July 4–8, 1994, Proceedings*, volume 817 of *Lecture Notes in Computer Science*, pages 37–48. Springer-Verlag, 1994.

[204] Patrick Lincoln and José Meseguer. Strategic reflection. In B. Gramlich and F. Pfenning, editors, *Proceedings of the CADE-15 Workshop on Strategies in Automated Deduction*, pages 3–9, Lindau, Germany, July 1998.

[205] Michael Lowry, Thomas Pressburger, and Grigore Roşu. Certifying domain-specific policies. Technical report, Research Institute for Advanced Computer Science, 2001.

[206] Dorel Lucanu. Algebraic specification of object aggregation — An event oriented approach. In Futatsugi et al. [140], pages 115–132.

[207] Dorel Lucanu. Relaxed models for rewriting logic. Manuscript, submitted for publication, July 2000.

[208] Narciso Martí-Oliet and José Meseguer. Rewriting logic as a logical and semantic framework. Technical Report SRI-CSL-93-05, SRI International, Computer Science Laboratory, August 1993. To appear in D. Gabbay, editor, *Handbook of Philosophical Logic, Second Edition, Volume 6*, Kluwer Academic Publishers, 2001. `http://maude.csl.sri.com/papers`.

[209] Narciso Martí-Oliet and José Meseguer. General logics and logical frameworks. In D. M. Gabbay, editor, *What is a Logical System?*, volume 4 of *Studies in Logic and Computation*, pages 355–392. Oxford University Press, 1994.

[210] Narciso Martí-Oliet and José Meseguer. Rewriting logic as a logical and semantic framework. In Meseguer [222], pages 189–224. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[211] Narciso Martí-Oliet and José Meseguer. Action and change in rewriting logic. In R. Pareschi and B. Fronhöfer, editors, *Dynamic Worlds: From the Frame Problem to Knowledge Management*, volume 12 of *Applied Logic Series*, pages 1–53. Kluwer Academic Publishers, 1999.

[212] Ian A. Mason and Carolyn L. Talcott. A semantics preserving actor translation. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 369–378. Springer-Verlag, 1997.

[213] Ian A. Mason and Carolyn L. Talcott. Simple network protocol simulation within Maude. In Futatsugi [139], pages 277–294. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[214] José Meseguer. A logical theory of concurrent objects. In N. Meyrowitz, editor, *Proceedings ECOOP-OOPSLA'90 Conference on Object-Oriented Programming, Ottawa, Canada, October 1990*, pages 101–115. ACM Press, 1990.

[215] José Meseguer. Rewriting as a unified model of concurrency. Technical Report SRI-CSL-90-02R, SRI International, Computer Science Laboratory, February 1990. Revised June 1990. Appendices on functorial semantics have not been published elsewhere.

[216] José Meseguer. Rewriting as a unified model of concurrency. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR'90, Theories of Concurrency: Unification and Extension, Amsterdam, The Netherlands, August 1990, Proceedings*, volume 458 of *Lecture Notes in Computer Science*, pages 384–400. Springer-Verlag, 1990.

[217] José Meseguer. Conditional rewriting logic: Deduction, models and concurrency. In S. Kaplan and M. Okada, editors, *Conditional and Typed Rewriting Systems, 2nd International CTRS Workshop, Montreal, Canada, June 11–14, 1990, Proceedings*, volume 516 of *Lecture Notes in Computer Science*, pages 64–91. Springer-Verlag, 1991.

[218] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[219] José Meseguer. Multiparadigm logic programming. In H. Kirchner and G. Levi, editors, *Algebraic and Logic Programming, Third International Conference, Volterra, Italy, September 2–4, 1992, Proceedings*, volume 632 of *Lecture Notes in Computer Science*, pages 158–200. Springer-Verlag, 1992.

[220] José Meseguer. A logical theory of concurrent objects and its realization in the Maude language. In G. Agha, P. Wegner, and A. Yonezawa, editors, *Research Directions in Concurrent Object-Oriented Programming*, pages 314–390. The MIT Press, 1993.

[221] José Meseguer. Solving the inheritance anomaly in concurrent object-oriented programming. In O. M. Nierstrasz, editor, *ECOOP'93 — Object-Oriented Programming, 7th European Conference, Kaiserslautern, Germany, July 26–30, 1993, Proceedings*, volume 707 of *Lecture Notes in Computer Science*, pages 220–246. Springer-Verlag, 1993.

[222] José Meseguer, editor. *Proceedings First International Workshop on Rewriting Logic and its Applications, WRLA'96, Asilomar, California, September 3–6, 1996*, volume 4 of *Electronic Notes in Theoretical Computer Science*. Elsevier, September 1996. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[223] José Meseguer. Rewriting logic as a semantic framework for concurrency: A progress report. In U. Montanari and V. Sassone, editors, *CONCUR'96: Concurrency Theory, 7th International Conference, Pisa, Italy, August 26–29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*, pages 331–372. Springer-Verlag, 1996.

[224] José Meseguer. Formal interoperability. In *Proceedings of the 1998 Conference on Mathematics in Artificial Intelligence*, Fort Laurerdale, Florida, January 1998. `http://rutcor.rutgers.edu/~amai/Proceedings.html`. Presented also at the *14th IMACS World Congress*, Atlanta, Georgia, July 1994.

[225] José Meseguer. A logical framework for distributed systems and communication protocols. In S. Budkowski, A. Cavalli, and E. Najm, editors, *Formal Description Techniques and Protocol Specification, Testing and Verification, FORTE/PSTV'98 IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XI) and Protocol Specification, Testing and Verification (PSTV XVIII), November 3–6, 1998, Paris, France*, volume 135 of *International Federation for Information Processing*, pages 327–333. Kluwer Academic Publishers, 1998.

[226] José Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Presicce, editor, *Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT'97, Tarquinia, Italy, June 3–7, 1997, Selected Papers*, volume 1376 of *Lecture Notes in Computer Science*, pages 18–61. Springer-Verlag, 1998.

[227] José Meseguer. Research directions in rewriting logic. In U. Berger and H. Schwichtenberg, editors, *Computational Logic, Proceedings of the NATO Advanced Study Institute on Computational Logic held in Marktoberdorf, Germany, July 29 – August 6, 1997*, volume 165 of *NATO ASI Series F: Computer and Systems Sciences*, pages 347–398. Springer-Verlag, 1998.

[228] José Meseguer. Rewriting logic and Maude: A wide-spectrum semantic framework for object-based distributed systems. In S. F. Smith and C. L.

Talcott, editors, *Proceedings IFIP Conference on Formal Methods for Open Object-Based Distributed Systems IV, FMOODS 2000, September 6–8, 2000, Stanford, California, USA*, pages 89–117. Kluwer Academic Publishers, 2000.

[229] José Meseguer. Rewriting logic and Maude: Concepts and applications. In L. Bachmair, editor, *Rewriting Techniques and Applications, 11th International Conference, RTA 2000, Norwich, UK, July 10–12, 2000, Proceedings*, volume 1833 of *Lecture Notes in Computer Science*, pages 1–26. Springer-Verlag, 2000.

[230] José Meseguer, Kokichi Futatsugi, and Timothy Winkler. Using rewriting logic to specify, program, integrate, and reuse open concurrent systems of cooperating agents. In *Proceedings of the 1992 International Symposium on New Models for Software Architecture*, pages 61–106, Tokyo, Japan, November 1992. Research Institute of Software Engineering.

[231] José Meseguer and Narciso Martí-Oliet. From abstract data types to logical frameworks. In E. Astesiano, G. Reggio, and A. Tarlecki, editors, *Recent Trends in Data Type Specification, 10th Workshop on Specification of Abstract Data Types Joint with the 5th COMPASS Workshop, S. Margherita, Italy, May 30 – June 3, 1994, Selected Papers*, volume 906 of *Lecture Notes in Computer Science*, pages 48–80. Springer-Verlag, 1995.

[232] José Meseguer and Ugo Montanari. Mapping tile logic into rewriting logic. In F. Parisi-Presicce, editor, *Recent Trends in Algebraic Development Techniques, 12th International Workshop, WADT'97, Tarquinia, Italy, June 3–7, 1997, Selected Papers*, volume 1376 of *Lecture Notes in Computer Science*, pages 62–91. Springer-Verlag, 1998.

[233] José Meseguer and Xiaolei Qian. A logical semantics for object-oriented databases. In P. Buneman and S. Jajodia, editors, *Proceedings International SIGMOD Conference on Management of Data*, pages 89–98. ACM, 1993.

[234] José Meseguer, Mark-Oliver Stehr, and Carolyn L. Talcott. Specifying the PLAN language in Maude, 2000. Slides available at `http://www-formal.stanford.edu/clt/Talks/00sep-utokyo-talk.ps.gz`.

[235] José Meseguer and Carolyn L. Talcott. Using rewriting logic to interoperate architectural description languages (I and II). Lectures at the Santa Fe and Seattle *DARPA-EDCS Workshops*, March and July 1997. `http://www-formal.stanford.edu/clt/ArpaNsf/adl-interop.html`.

[236] José Meseguer and Carolyn L. Talcott. Mapping OMRS to rewriting logic. In Kirchner and Kirchner [167], pages 345–366. `http://www.elsevier.nl/locate/entcs/volume15.html`. Full version in preparation.

[237] José Meseguer and Carolyn L. Talcott. A partial order event model for concurrent objects. In J. C. M. Baeten and S. Mauw, editors, *CONCUR'99, Concurrency Theory, 10th International Conference Eindhoven, The Netherlands, August 24–27, 1999, Proceedings*, volume 1664 of *Lecture Notes in Computer Science*, pages 415–430. Springer-Verlag, 1999.

[238] José Meseguer and Timothy Winkler. Parallel programming in Maude. In J.-P. Banâtre and D. Le Mètayer, editors, *Research Directions in High-level Parallel Programming Languages, Mont Saint-Michel, France, June 17–19, 1991, Proceedings*, volume 574 of *Lecture Notes in Computer Science*, pages 253–293. Springer-Verlag, 1992.

[239] Jon Millen. Applications of term rewriting to cryptographic protocol analysis. In Futatsugi [139], pages 229–234. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[240] Hiroyuki Miyoshi. Modelling conditional rewriting logic in structured categories. In Meseguer [222], pages 20–34. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[241] Ugo Montanari and Carolyn L. Talcott. Can actors and pi-agents live together? In A. D. Gordon, A. M. Pitts, and C. L. Talcott, editors, *Proceedings Second Workshop on Higher-Order Operational Techniques in Semantics, HOOTS'98*, volume 10 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1998.

[242] Pierre-Etienne Moreau. A choice-point library for backtrack programming. In *JICSLP'98 Post-Conference Workshop on Implementation Technologies for Programming Languages based on Logic*, June 1998.

[243] Pierre-Etienne Moreau. *Compilation de Règles de Réécriture et de Stratégies Non-Déterministes*. PhD thesis, Université Henri Poincaré – Nancy I, June 1999.

[244] Pierre-Etienne Moreau and Hélène Kirchner. Compilation of associative-commutative normalisation with strategies in ELAN (full version). Technical Report 97-R-129, LORIA, Nancy, France, 1997.

[245] Pierre-Etienne Moreau and Hélène Kirchner. Compilation techniques for associative-commutative normalisation. In M. P. A. Sellink, editor, *Second International Workshop on the Theory and Practice of Algebraic Specifications, Amsterdam, The Netherlands, September 25–26, 1997*, Electronic Workshops in Computing. Springer-Verlag, 1998. `http://www.ewic.org.uk/ewic/workshop/view.cfm/ASFSDF-97`.

[246] Pierre-Etienne Moreau and Hélène Kirchner. A compiler for rewrite programs in associative-commutative theories. In C. Palamidessi, H. Glaser, and K. Meinke, editors, *Principles of Declarative Programming, 10th International Symposium, PLIP'98 Held Jointly with the 6th Conference ALP'98, Pisa, Italy, September 16–18, 1998, Proceedings*, volume 1490 of *Lecture Notes in Computer Science*, pages 230–249. Springer-Verlag, 1998.

[247] Peter D. Mosses. Semantics, modularity, and rewriting logic. In Kirchner and Kirchner [167]. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[248] Peter D. Mosses. Logical specification of operational semantics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic, 13th International Workshop, CSL'99, 8th Annual Conference of the EACSL,*

Madrid, Spain, September 20–25, 1999, Proceedings, volume 1683 of *Lecture Notes in Computer Science*, pages 32–49. Springer-Verlag, 1999.

[249] Elie Najm and Jean-Bernard Stefani. A formal operational semantics for the ODP computational model with signals, explicit binding, and reactive objects. Manuscript, ENST, Paris, France, 1994.

[250] Elie Najm and Jean-Bernard Stefani. A formal semantics for the ODP computational model. *Computer Networks and ISDN Systems*, 27:1305–1329, 1995.

[251] Elie Najm and Jean-Bernard Stefani. Computational models for open distributed systems. In H. Bowman and J. Derrick, editors, *Proceedings Second IFIP Conference on Formal Methods for Open Object-Based Distributed Systems, FMOODS'97, July 21–23, 1997, Canterbury, Kent, UK*, pages 157–176. Chapman & Hall, 1997.

[252] Shin Nakajima. Encoding mobility in CafeOBJ: An exercise of describing mobile code-based software architecture. In Futatsugi [138].

[253] Shin Nakajima. Using algebraic specification techniques in development of object-oriented frameworks. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 — Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, September 20–24, 1999 Proceedings, Volume II*, volume 1709 of *Lecture Notes in Computer Science*, pages 1664–1683. Springer-Verlag, 1999.

[254] Shin Nakajima and Kokichi Futatsugi. An object-oriented modeling method for algebraic specifications in CafeOBJ. In *Proceedings, 19th International Conference on Software Engineering*, pages 34–44, Boston, Massachussets, May 1997. IEEE Computer Society Press.

[255] Masaki Nakamura and Kazuhiro Ogata. The evaluation strategy for head normal form with and without on-demand flags. In Futatsugi [139], pages 211–227. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[256] Hideyuki Nakashima. Cyber assistance for situated human information processing. In Futatsugi [139], pages 295–296. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[257] Pavel Naumov, Mark-Oliver Stehr, and José Meseguer. The HOL/NuPRL proof translator — A practical approach to formal interoperability. In *Theorem Proving in Higher Order Logics, 14th International Conference, TPHOLs'2001, Edinburgh, Scotland, UK, September 3–6, 2001, Proceedings*, Lecture Notes in Computer Science. Springer-Verlag, 2001.

[258] Thomas Noll. On coherence properties in term rewriting models of concurrency. In J. C. M. Baeten and S. Mauw, editors, *CONCUR'99, Concurrency Theory, 10th International Conference Eindhoven, The Netherlands, August 24–27, 1999, Proceedings*, volume 1664 of *Lecture Notes in Computer Science*, pages 478–493. Springer-Verlag, 1999.

[259] Masanobu Numazawa, Masahito Kurihara, and Azuma Ohuchi. A reflective language based on conditional term rewriting. Technical report, Division of Systems and Information Engineering, Hokkaido University, Sapporo, Japan, 1996.

[260] Kazuhiro Ogata and Kokichi Futatsugi. An abstract machine for order-sorted conditional term rewriting systems. In H. Comon, editor, *Rewriting Techniques and Applications, 8th International Conference, RTA'97, Sitges, Spain, June 2–5, 1997, Proceedings*, volume 1232 of *Lecture Notes in Computer Science*, pages 335–338. Springer-Verlag, 1997.

[261] Kazuhiro Ogata and Kokichi Futatsugi. Specification and verification of some classical mutual exclusion algorithms with CafeOBJ. In Futatsugi et al. [140], pages 159–178.

[262] Peter Csaba Ölveczky. *Specification and Analysis of Real-Time and Hybrid Systems in Rewriting Logic*. PhD thesis, University of Bergen, Norway, 2000. `http://maude.csl.sri.com/papers`.

[263] Peter Csaba Ölveczky, Mark Keaton, José Meseguer, Carolyn L. Talcott, and Steve Zabele. Specification and analysis of the AER/NCA active network protocol suite in Real-Time Maude. In H. Hussmann, editor, *Fundamental Approaches to Software Engineering, 4th International Conference, FASE 2001, Held as Part of ETAPS 2001, Genova, Italy, April 2001, Proceedings*, volume 2029 of *Lecture Notes in Computer Science*, pages 333–347. Springer-Verlag, 2001. `http://maude.csl.sri.com/papers`.

[264] Peter Csaba Ölveczky, Piotr Kosiuczenko, and Martin Wirsing. An object-oriented algebraic steam-boiler control specification. In J.-R. Abrial, E. Börger, and H. Langmaack, editors, *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, volume 1165 of *Lecture Notes in Computer Science*, pages 379–402. Springer-Verlag, 1996.

[265] Peter Csaba Ölveczky and Sigurd Meldal. Specification and prototyping of network protocols in rewriting logic. In *Proceedings of NIK'98, Norsk Informatikk Konferanse*, 1998.

[266] Peter Csaba Ölveczky and José Meseguer. Specifying real-time systems in rewriting logic. In Meseguer [222], pages 283–308. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[267] Peter Csaba Ölveczky and José Meseguer. Real-Time Maude: A tool for simulating and analyzing real-time and hybrid systems. In Futatsugi [139], pages 361–383. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[268] Peter Csaba Ölveczky and José Meseguer. Specification of real-time and hybrid systems in rewriting logic. *Theoretical Computer Science*, 2001. This volume.

[269] Peter Csaba Ölveczky and José Meseguer. Specifying and analyzing real-time object systems in Real-Time Maude. Manuscript, Computer Science Laboratory, SRI International, submitted for publication, 2001.

[270] Miguel Palomino Tarjuelo. Comparing Meseguer's rewriting logic with the logic CRWL. In *Proceedings WFLP 2001, International Workshop on Functional and (Constraint) Logic Programming*, Kiel, Germany, September 13–15, 2001.

[271] Miguel Palomino Tarjuelo. Relating Meseguer's rewriting logic with the constructor-based rewriting logic. Master's thesis, Facultad de Matemáticas, Universidad Complutense de Madrid, Spain, May 2001. `http://maude.csl.sri.com/papers`.

[272] Dirk Pattinson. Modal logic for rewriting theories. In Futatsugi [139], pages 173–191. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[273] Isabel Pita and Narciso Martí-Oliet. A Maude specification of an object oriented database model for telecommunication networks. In Meseguer [222], pages 404–422. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[274] Isabel Pita and Narciso Martí-Oliet. Using reflection to specify transaction sequences in rewriting logic. In J. L. Fiadeiro, editor, *Recent Trends in Algebraic Development Techniques, 13th International Workshop, WADT'98, Lisbon, Portugal, April 2–4, 1998, Selected Papers*, volume 1589 of *Lecture Notes in Computer Science*, pages 261–276. Springer-Verlag, 1999.

[275] Isabel Pita and Narciso Martí-Oliet. A Maude specification of an object-oriented model for telecommunication networks. *Theoretical Computer Science*, 2001. This volume.

[276] José F. Quesada. The Maude parser: Parsing and meta-parsing β-extended context-free grammars. Technical report, Computer Science Laboratory, SRI International, 2001. To appear.

[277] Christophe Ringeissen. Prototyping combination of unification algorithms with the ELAN rule-based programming language. In H. Comon, editor, *Rewriting Techniques and Applications, 8th International Conference, RTA'97, Sitges, Spain, June 2–5, 1997, Proceedings*, volume 1232 of *Lecture Notes in Computer Science*, pages 323–326. Springer-Verlag, 1997.

[278] Christophe Ringeissen. Handling relations over finite domains in the rule-based system ELAN. In Futatsugi [139], pages 193–210. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[279] Dilia E. Rodríguez. Case studies in the specification and analysis of protocols in Maude. In Futatsugi [139], pages 257–275. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[280] Grigore Roşu and Klaus Havelund. Generating optimal monitors from temporal formulae. Technical report, Research Institute for Advanced Computer Science, 2001.

[281] Barbara Salmansberger. Objektorientierte Spezifikation von verteilten Systemen in Maude am Beispiel eines Flughafens. Master's thesis, Fakultät für Mathematik und Informatik, Universität Passau, December 1993.

[282] Marisol Sánchez, José Luis Herrero, Juan Manuel Murillo, and Juan Hernández. Guaranteing coherent software systems when composing coordinated components. In A. Porto and G.-C. Roman, editors, *Coordination Languages and Models, 4th International Conference, COORDINATION 2000, Limassol, Cyprus, September 11–13, 2000, Proceedings*, volume 1906 of *Lecture Notes in Computer Science*, pages 341–346. Springer-Verlag, 2000.

[283] Christelle Scharff. *Déduction avec Contraintes et Simplification dans les Théories Équationnelles*. PhD thesis, Université Henri Poincaré – Nancy I, September 1999.

[284] W. Marco Schorlemmer. Bi-rewriting rewriting logic. In Meseguer [222], pages 265–282. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[285] W. Marco Schorlemmer. Rewriting logic as a logic of special relations. In Kirchner and Kirchner [167], pages 163–184. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[286] W. Marco Schorlemmer. *On Specifying and Reasoning with Special Relations*. PhD thesis, Universitat Politècnica de Catalunya, March 1999.

[287] W. Marco Schorlemmer. Term rewriting in a logic of special relations. In A. M. Haeberer, editor, *Algebraic Methodology and Software Technology, 7th International Conference, AMAST'98, Amazonia, Brazil, January 4–8, 1999, Proceedings*, volume 1548 of *Lecture Notes in Computer Science*, pages 178–195. Springer-Verlag, 1999.

[288] David B. Skillicorn and Domenico Talia. Models and languages for parallel computation. *ACM Computing Surveys*, 30(2):123–169, June 1998.

[289] L. Jason Steggles. Rewriting logic and Elan: Prototyping tools for Petri nets with time. In J.-M. Colom and M. Koutny, editors, *Applications and Theory of Petri Nets 2001, 22nd International Conference, ICATPN 2001, Newcastle upon Tyne, UK, June 25–29, 2001, Proceedings*, volume 2075 of *Lecture Notes in Computer Science*, pages 363–381. Springer-Verlag, 2001.

[290] L. Jason Steggles and Piotr Kosiuczenko. A timed rewriting logic semantics for SDL: A case study of the alternating bit protocol. In Kirchner and Kirchner [167], pages 295–316. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[291] L. Jason Steggles and Piotr Kosiuczenko. A formal model for SDL specifications based on timed rewriting logic. *Automated Software Engineering*, 7(1):61–90, March 2000.

[292] Mark-Oliver Stehr. CINNI — A generic calculus of explicit substitutions and its application to λ-, ς- and π-calculi. In Futatsugi [139], pages 71–92. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[293] Mark-Oliver Stehr. A rewriting semantics for algebraic nets. In C. Girault and R. Valk, editors, *Petri Nets for System Engineering — A Guide to Modeling, Verification, and Applications*. Springer-Verlag, 2001.

[294] Mark-Oliver Stehr. *Rewriting Logic and Type Theory — From Applications to Unification.* PhD thesis, Computer Science Department, University of Hamburg, Germany, 2002. In preparation.

[295] Mark-Oliver Stehr and José Meseguer. Pure type systems in rewriting logic. In *Proceedings of LFM'99: Workshop on Logical Frameworks and Meta-languages*, Paris, France, September 28, 1999. `http://www.cs.bell-labs.com/~felty/LFM99/`.

[296] Mark-Oliver Stehr, José Meseguer, and Peter Csaba Ölvczky. Representation and execution of Petri nets using rewriting logic as a uniform framework. In J. Padberg, editor, *Proceedings UNIGRA'2001, Uniform Approaches to Graphical Process Specification Techniques*, Electronic Notes in Theoretical Computer Science, Genova, Italy, March/April 2001. Elsevier. To appear.

[297] Mark-Oliver Stehr, José Meseguer, and Peter Csaba Ölveczky. Rewriting logic as a unifying framework for Petri nets. In H. Ehrig, G. Juhas, J. Padberg, and G. Rozenberg, editors, *Unifying Petri Nets*, Lecture Notes in Computer Science. Springer-Verlag, 2001. To appear.

[298] Mark-Oliver Stehr, Pavel Naumov, and José Meseguer. A proof-theoretic approach to the HOL-Nuprl connection with applications to proof translation. In M. Cerioli, P. D. Mosses, and G. Reggio, editors, *Proceedings WADT/CoFI'01, 15th International Workshop on Algebraic Development Techniques and General Workshop of the CoFI WG*, Genova, Italy, April 1–3, 2001. `http://www.csl.sri.com/~stehr/fi_eng.html`.

[299] Yasuyuki Tahara, Fumihiro Kumeno, Akihiko Ohsuga, and Shinichi Honiden. An algebraic semantics of reflective objects. In K. Futatsugi and S. Matsuoka, editors, *Object-Technologies for Advanced Software, Second JSSST International Symposium, ISOTAS'96, Kanaza, Japan, March 11–15, 1996, Proceedings*, volume 1049 of *Lecture Notes in Computer Science*, pages 173–189. Springer-Verlag, 1996.

[300] Carolyn L. Talcott. An actor rewriting theory. In Meseguer [222], pages 360–383. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[301] Carolyn L. Talcott. Composable semantic models for actor theories. In M. Abadi and T. Ito, editors, *Theoretical Aspects of Computer Science, Third International Symposium, TACS'97, Sendai, Japan, September 23–26, 1997, Proceedings*, volume 1281 of *Lecture Notes in Computer Science*, pages 321–364. Springer-Verlag, 1997.

[302] Carolyn L. Talcott. Interaction semantics for components of distributed systems. In E. Najm and J.-B. Stefani, editors, *Proceedings IFIP Conference on Formal Methods for Open Object-Based Distributed Systems, FMOODS'96*, pages 154–169. Chapman & Hall, 1997.

[303] Carolyn L. Talcott. Towards a toolkit for actor system specification. In T. Rus, editor, *Algebraic Methodology and Software Technology, 8th International Conference, AMAST 2000, Iowa City, Iowa, USA, May 20–27,*

2000, Proceedings, volume 1816 of *Lecture Notes in Computer Science*, pages 391–406. Springer-Verlag, 2000.

[304] Carolyn L. Talcott. Actor theories in rewriting logic. *Theoretical Computer Science*, 2001. This volume.

[305] Ambrosio Toval and José Luis Fernández. Formally modeling UML and its evolution: A holistic approach. In S. F. Smith and C. L. Talcott, editors, *Proceedings IFIP Conference on Formal Methods for Open Object-Based Distributed Systems IV, FMOODS 2000, September 6–8, 2000, Stanford, California, USA*, pages 183–206. Kluwer Academic Publishers, 2000.

[306] Ambrosio Toval and José Luis Fernández. Improving system reliability via rigorous software modeling: The UML case. In *Proceedings IEEE Aerospace Conference, Volume 6*, pages 6–17, Big Sky, MT, USA, March 10–17, 2001. IEEE Press.

[307] Mark G. J. van den Brand, Paul Klint, and Chris Verhoef. Term rewriting for sale. In Kirchner and Kirchner [167], pages 139–162. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[308] Mark G. J. van den Brand and Christophe Ringeissen. ASF+SDF parsing tools applied to ELAN. In Futatsugi [139], pages 135–154. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[309] Alberto Verdejo and Narciso Martí-Oliet. Executing E-LOTOS processes in Maude. In H. Ehrig, M. Grosse-Rhode, and F. Orejas, editors, *INT 2000, Integration of Specification Techniques with Applications in Engineering, Extended Abstracts*, pages 49–53. Technical report 2000/04, Technische Universitat Berlin, March 2000.

[310] Alberto Verdejo and Narciso Martí-Oliet. Executing and verifying CCS in Maude. Technical Report 99-00, Departamento de Sistemas Informáticos y Programación, Universidad Complutense de Madrid, February 2000. `http://maude.csl.sri.com/casestudies/ccs`.

[311] Alberto Verdejo and Narciso Martí-Oliet. Implementing CCS in Maude. In T. Bolognesi and D. Latella, editors, *Formal Methods For Distributed System Development. FORTE/PSTV 2000 IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XIII) and Protocol Specification, Testing and Verification (PSTV XX) October 10–13, 2000, Pisa, Italy*, volume 183 of *International Federation for Information Processing*, pages 351–366. Kluwer Academic Publishers, 2000.

[312] Alberto Verdejo, Isabel Pita, and Narciso Martí-Oliet. The leader election protocol of IEEE 1394 in Maude. In Futatsugi [139], pages 385–406. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[313] Patrick Viry. *La Réécriture Concurrente.* PhD thesis, Université de Nancy I, 1992.

[314] Patrick Viry. Rewriting: An effective model of concurrency. In C. Halatsis, D. Maritsas, G. Philokyprou, and S. Theodoridis, editors, *PARLE'94 Parallel Architectures and Languages Europe, 6th International PARLE Conference, Athens, Greece, July 4–8, 1994, Proceedings*, volume 817 of *Lecture Notes in Computer Science*, pages 648–660. Springer-Verlag, 1994.

[315] Patrick Viry. Rewriting modulo a rewrite system. Technical Report TR-95-20, Dipartimento di Informatica, Università di Pisa, December 1995. `ftp://ftp.di.unipi.it/pub/techreports/TR-95-20.ps.Z`.

[316] Patrick Viry. Input/output for ELAN. In Meseguer [222], pages 51–64. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[317] Patrick Viry. Adventures in sequent calculus modulo equations. In Kirchner and Kirchner [167], pages 367–378. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[318] Patrick Viry. Equational rules for rewriting logic. *Theoretical Computer Science*, 2001. This volume.

[319] Eelco Visser and Zine el Abidine Benaissa. A core language for rewriting. In Kirchner and Kirchner [167], pages 25–44. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[320] Marian Vittek. *ELAN: Un Cadre Logique pour le Prototypage de Langages de Programmation avec Contraintes*. PhD thesis, Université Henri Poincaré – Nancy I, November 1994.

[321] Marian Vittek. A compiler for nondeterministic term rewriting systems. In H. Ganzinger, editor, *Rewriting Techniques and Applications, 7th International Conference, RTA'96, New Brunswick, NJ, USA July 27–30, 1996, Proceedings*, volume 1103 of *Lecture Notes in Computer Science*, pages 154–168. Springer-Verlag, 1996.

[322] Bow-Yaw Wang, José Meseguer, and Carl A. Gunter. Specification and formal analysis of a PLAN algorithm in Maude. In P.-A. Hsiung, editor, *Proceedings International Workshop on Distributed System Validation and Verification, Taipei, Taiwan*, pages 49–56, April 2000.

[323] Takuo Watanabe. Towards a foundation of computational reflection based on abstract rewriting (preliminary result). In *Proceedings IMSA'95*, pages 143–145. Information-Technology Promotion Agency, Japan, 1995.

[324] Takuo Watanabe, Hiroshi Ishikawa, and Kokichi Futatsugi. Towards declarative description of computational reflection. In *Proceedings IMSA'96*, pages 113–128. Information-Technology Promotion Agency, Japan, 1996.

[325] Timothy Winkler. Programming in OBJ and Maude. In P. Lauer, editor, *Functional Programming, Concurrency, Simulation and Automated Reasoning, International Lecture Series 1991–1992, McMaster University, Hamilton, Ontario, Canada*, volume 693 of *Lecture Notes in Computer Science*, pages 229–277. Springer-Verlag, 1993.

[326] Martin Wirsing and Alexander Knapp. A formal approach to object-oriented software engineering. In Meseguer [222], pages 321–359. `http://www.elsevier.nl/locate/entcs/volume4.html`.

[327] Martin Wirsing and Alexander Knapp. A formal approach to object-oriented software engineering. *Theoretical Computer Science*, 2001. This volume.

[328] Martin Wirsing, Friederike Nickl, and Ulrike Lechner. Concurrent object-oriented specification in SPECTRUM. In Y. Inagaki, editor, *Workshop on algebraic and Object-Oriented Approaches to Software Science*, pages 39–70, Nagoya, Japan, 1995.