Combination and Interoperation of Logical Systems
Research in Formal Interoperability

Final report for ONR grant NAVY N00014-94-1-0857
06/01/94 - 04/30/99

1 Project Objectives

This project began with the long term objective of being able to hook together reasoning systems in semantically meaningful and useful ways, where the term reasoning system covers a wide range of tools including stand-alone theorem provers, decision procedures, and tools for transforming or translating formal assertions. The problem of combining and interoperating theorem provers and logical systems provides a very good focus to fruitfully apply, and to further develop existing formal theories such as: general logics; rewriting logic; module calculi; and meta architectures for reasoning systems. The theory of general logics [Mes89] provides rigorous criteria of correctness and semantics for the activity of "hooking proof systems together". Rewriting logic [Mes92, Mes98] is a logical framework [MOM97], or "universal logic" in which other logics can naturally be represented by means of maps of logics. The modularity and parameterization facilities of rewriting logic lead immediately to modularity and parameterization of presentations.

Going beyond reasoners we want to interoperate different formalisms and systems based on diverse computational models. Thus, this project involved investigation of both interpretations of the phrase *formal interoperability*: (i) formal semantics of the interoperation of components and their combination into complex systems; and (ii) the interoperation of formal systems and their combined use to specify the many aspects of complex systems. The research addressed needs for: (i) scientific foundations for sensible, correct and secure interaction between components; (ii) formalizing different aspects of complex systems and reasoning across such formalizations; and (iii) correct interoperation of formal tools such as theorem provers, declarative languages, and analyzers.

By using reflective techniques and taking an even more abstract point of view of module compositionality in which module composition operators apply to both formal and informal module descriptions as well as to the software resources that implement the modules, a module calculus can deal simultaneously with all the aspects of a resource. Such a module calculus can be of great practical use in the design, development, evolution and reuse of software systems and can be realized in highly generic tools of wide applicability. The Maude language and interpreter have been used to implement some of the formal interoperability ideas. Maude is a fully reflective and very efficient implementation of rewriting logic that naturally supports such efforts. Maude executables, its manual, and a collection of examples and papers are available on the Web (`http://maude.csl.sri.com`).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	99 November 29	Final Report 06/01/94 - 04/30/99

4. TITLE AND SUBTITLE
Combination and Interoperation of Logical Systems
Research in Formal Interoperability

5. FUNDING NUMBERS
C:
ONR N00014-94-1-0857

6. AUTHOR(S)
John McCarthy/
Carolyn Talcott

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computer Science Department
353 Serra Mall
Gates Building, 2A
Stanford University
Stanford, CA 94305-9020

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Administrative Grants Officer
OFFICE OF NAVAL RESEARCH REGIONAL OFFICE - SEATTLE
1107 NE 45th Street Suite 350
Seattle, WA 98105-4631

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This project involved investigation of both interpretations of the phrase formal interoperability: (i) formal semantics of the interoperation of components and their combination into complex systems; and (ii) the interoperation of formal systems and their combined use to specify the many aspects of complex systems.

(i) scientific foundations for sensible, correct and secure interaction between components;
(ii) formalizing different aspects of complex systems and reasoning across such formalizations; and
(iii) correct interoperation of formal tools such as theorem provers, declarative languages, and analyzers.

Progress in several areas of formal interoperability is reported: reasoning systems, heterogeneous architectures, distributed open systems and the use of reflective techniques.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Formal Interoperability, module calculus, reasoning theory, reflection	9
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

2 Technical Results

Progress has been made in several areas of formal interoperability: reasoning systems, heterogeneous architectures, distributed open systems, and the use of reflective techniques. Results are described in the following subsections. In addition we established and continue to maintain a mechanized reasoning systems web site that include a database of existing reasoning systems and other related information. In 1996 we merged our site with a site maintained at the University of Saarbrüken in Germany. This resulted in added content and better accessibility. The main page can be accessed at `http://www-formal.stanford.edu/clt/ARS/ars-db.html`.

2.1 Open Mechanized Reasoning Systems

The Open Mechanized Reasoning Systems (OMRS) Framework was developed as a joint effort between this project and the Mechanized Reasoning Group at IRST (Trento Italy) and the University of Genova (Genova, Italy). (The web page for the OMRS project is `http://www.mrg.dist.unige.it/omrs/index.html`.)

The notion of an *Open Mechanized Reasoning System* (OMRS) was introduced [GPT96] as a framework for specifying mechanized reasoning systems. The long term objective of the OMRS project is to provide a technology for integrating and interoperating diverse proof systems with each other and with other software components (for example symbolic algebra systems, compilers, information systems) to build complex systems from existing and newly created components in a principled and sound manner, with minimal changes to the existing modules. An OMRS has three aspects: a reasoning theory specifying the underlying inference system; strategies for controlling inference; and interaction capabilities.

Reasoning theories [GPT96] have several aspects: sequents (assertions), constraint systems (for expressing and checking applicability conditions), instantiation systems (for instantiating schematic entities such as sequents and constraints) and rules for deduction. The semantics of a reasoning theory is given by structures called *reasoning structures* – structures that represent proof fragments and that naturally support the process of constructing a proof, automatically and/or interactively. We have developed a rich variety of algebraic and categorical structure for reasoning theories and the associated structures, guided by the analogy to rewriting logic [MT].

Reasoning theories come in several flavors: abstract, nested and equationally presented. *Abstract reasoning theories (Arths)* treat sequents, constraints and instantiations axiomatically. Constraint systems form symmetric, strict monoidal categories with weak terminal objects. Instantiation systems are monoids of instantiation maps and sequent and constraint systems come equipped with structure preserving actions of the instantiation monoid. Mappings between abstract reasoning theories and between the various component systems turns these collections of structures into categories. Nested reasoning theories (NRThs) have constraint systems that are presented as (Nested) reasoning theories. Equationally presented reasoning theories (ERThs), have sequents presented by equational theories. In

an ERTh, the instantiation monoid is generated uniformly by extending the sequent algebra with meta variables and substitutions – using freeness of the algebra. Each NRTh, and ERth has an associated underlying ARTh.

To each Arth is associated a set of reasoning structures (proof fragments) and composable reasoning structures. Reasoning structures are associated to NRThs and ERths via the mapping to Arths. This association is functorial. Composable reasoning structures have a 2-categorical structure with underlying categorical structure essentially that of constraint systems. There is an algebra of arrows, analogous to the algebra of proofs in a rewriting theory, that provides for several dimensions of modularity and notions of composition of reasoning structures. These algebras provide a basis for specifying inference modules in a reasoning system. We have discovered a number of connections to Petri nets considered as special kinds of rewriting theories: the notion of occurrence net has an analog in reasoning structures as those structures in which sequent nodes are not shared; and composable processes of the Petri net theory are analogous to composable reasoning structures. Certain equations in the algebra of reasoning structures that correspond to the equational theory of rewriting proof terms, equate graphs that are in general different if the structure of a proof matters. A similar phenomenon arises axiomatizing the algebra of net computations.

In [GPT94] another algebra of reasoning structures was proposed. That algebra is based on the operations needed to support dynamic, interactive exploration and proof construction. It manipulates reasoning structures that may have gaps and hence are not necessarily derivation structures. Both algebras are important for supporting the design and interoperation off mechanized reasoning systems.

There is an obvious intuitive analogy between reasoning theories and rewriting logic: sequents play the role of terms in a rewriting theory; constraints play a role analogous to conditions in a conditional rewrite rule; reasoning theory rules correspond to the rule part of a rewriting logic theory; and the algebra of composable reasoning structures corresponds to the algebra of proof terms that form the initial model of a rewriting theory. This analogy has been formalized as a mapping of logics from ERThs to a special class of theories of rewriting logic [MT98].

One of the objectives of our study was to provide a formal notion of modularity and composition of Reasoning Theories. This has been outlined for ERThs using the notion of faithful inclusion mapping between ERTHs [CGMT98]. In the process we have defined a new notion we call 'strong persistence' which characterizes the embedding of the theory shared by the constraint and sequent systems in a reasoning theory [MT]. Pushouts preserve strong persistence, thus they can be used to describe the combination of a sequent algebra and a constraint algebra into a reasoning theory – using the uniform construction of instantiation monoids as substitution systems.

The reasoning theory framework has been used to analyze the NQTHM prover [CGPT97] and to develop an modular reconstruction of the high-level structure of the ACL2 prover [Ber97]. The latter augments ACL2 with the ability to construct proofs. In addition the reasoning theory framework has been used as the basis of an experiment to re-engineer the integration of the NQTHM linear arithmetic procedure implemented as a re-usable

separate process [AR97]. A formalism for the control component of OMRS was developed and used to specify the control component of NQTHM [Cog96] The OMRS framework has been used to build provably correct systems [GPA96], and it has be used as the basis for integration of the proof systems and symbolic algebra systems [BCGH98].

Important progress has also been made in developing techniques for reusing logics, so that both formal systems and their computer implementations can be used not only in their original context, but also across different logics by a process called "borrowing" [CM97a]. In such a process, a map relating part of the logical structures of two logics can be used to borrow the remaining logical structure from the target logic by a generalized process of "pulling it back" along the given map. For example, formal reasoning techniques and system implementations for equational logic can be borrowed in a sound and complete way to do reasoning in linear logic in an equational style, instead of in the usual sequent style. General borrowing results as well as the conditions under which those results specialize to subcases enjoying particularly nice conservation of logical properties have been studied using the general logics framework. It turns out that these results can be given a particularly simple and elegant formulation in terms of fibered categories. The constructions are illustrated with relevant examples that show how the techniques can be applied in a wide variety of situations.

2.2 Heterogeneous Architectures

A substantial case study showing how rewriting logic Maude can be used to execute very high level software designs, namely, architectural descriptions, has been carried out ([CDE+98] (Appendix E). It focuses on a difficult case, namely, *heterogeneous* architectures, illustrated by a command-and-control example featuring dataflow, message passing, and implicit invocation subarchitectures. Using Maude, each of the different subarchitectures cannot only be executed, but can also be interoperated in the execution of the resulting overall system.

2.3 Distributed Object-based Open Systems

In addition to investigating the connection between rewriting logic and reasoning theories, we have been investigating the use of rewriting logic as a semantic framework for open and highly concurrent systems, including actor systems, and concurrent object-oriented systems. In [Tal96b] an abstract operational semantic framework, *Actor Theories* (called abstract actor structures there) for specifying actor systems was introduced along with a more abstract trace-like model called interaction semantics. Actor theories can be thought of as specifying the interface between individual actor programs and the runtime environment that supports their execution and interaction. This work provides a first step in the development of a semantics of distributed and interoperable components open systems. It has already served as the basis for a temporal logic specification formalism for actor systems [Dua99] and for proving correctness of translations between actor languages [MT99a].

In [Tal96a] the rewriting logic foundations of actor theories is formalized by defining the notion of an Actor rewrite theory and developing an algebra of actor system computations based on this. In [Tal99] the notion of equationally presented actor theories (EPATs) is defined along with a theory mapping from EPATs to theories of rewriting logic. This mapping is the basis for an implementation of actor systems in Maude. Extending this mapping to the infinite fair computations of actor theories adds a new dimension to the models of rewriting logic theories of concurrent objects.

Rewriting logic directly provides an operational semantic model for actor theories with rich algebraic structure. Generalizing this structure, a category of algebras of actor system components has been developed. Actor configurations, computation paths, event-diagrams (a partial-order of events model), and interaction paths have all been given actor algebra structure. Morphism of the category provide composable semantics and means for transferring results between different models [Tal97, Tal98]. A graphical notation for specifying actor systems has been developed and given semantics using such morphisms [ST99].

A a general partial order of events model for object-based Maude specifications has been developed. This model coincides, for finite computations, with the event-diagram model of actor systems. It provides a semantic basis for interoperation of components described in languages with different underlying computation models [MT99b].

2.4 Reflection

Progress has been made in both logical reflection and computational reflection.

In collaboration with Manuel Clavel and axiomatization of reflective logics within the theory of general logics has been developed [CM96a]. The key concept is that of a universal theory which can simulate all the finitary theories in a given class of theories. This concept has many important applications, first of all to the rigorous the design of reflective declarative languages and to modularity and metaprogramming methodology. Another important application is a new declarative approach to deductive strategies in declarative languages and theorem-provers [CM97b]. Up to now, strategies have usually been defined by some kind of tactical language separated for the underlying logic. What reflection permits is to make strategies internal to the logic whose deductions they control. This can be achieved in an elegant way using reflection, and has been already illustrated in practice for the Maude rewriting logic language [CM96b, Cla98]. An extensible module algebra with powerful composition operations has been designed and implemented using reflective techniques in the Full Maude language [Dur99]. This is a first step towards developing a logic-independent generic module algebra applicable to a wide range of formal notations [DM98].

Two reflective architectures for actor computation have been used to provide a basis for defining and reasoning about composable services in dynamic adaptable distributed systems: the *onion skin* and the *two-level actor model.* The *onion skin* or layered architecture aims at modular specification of interaction policies for components of

distributed systems. It provides a simple means of structuring interaction policies in layers and allows for policy enforcers to be dynamically installed and to cleanly disappear when no longer needed. The reflective model underlying the onion skin architecture provides a means of modifying communication and state of actors on an individual basis. In this model each actor has a single meta-actor that can observe and modify its behavior. The default meta-actor implements the standard semantics. Behavior is modified by explicitly installing a meta-actor. Since the result of installing a meta-actor is again an actor, this process can be repeated indefinitely. This allows different protocols for fault-tolerance, security, reliability, quality of service, and other policies to be composed and modified dynamically as the system runs and new needs arise by adding new meta-levels. It has been used to support a number of high-level declarative programming abstractions such as synchronizers, activators, real-time synchronizers, actor spaces, and protocols that abstract over interaction patterns. We have defined an executable rewriting logic semantics for distributed object-oriented systems, based on the onion skin model, that explicitly addresses the reflective properties that are essential for having a truly modular notion of communication service. This semantics has been used in a case study treating communication services such as fault-tolerance, encryption and authentication [DMT99].

The *two level actor model* (TLAM) provides mechanisms for describing distributed system services built from services local to a node. Here each node has a group of base-level (application) actors and a group of meta-level actors that observe and control the base-level behavior. Meta-level actors from different nodes in a network interact to provide distributed systemwide services. This model has been used to formalize and reason about a distributed garbage collection algorithm [VAT92] and the safe composition of system-level activities such as remote creation, migration, and recording of global snapshots of system properties such as accessibility relation between actors [VT93, VT95]. It has also been used to model and reason about an architecture for quality-of-service-based multimedia services [Ven98, VAT99].

References

[AR97] A. Armando and S. Ranise. From integrated reasoning specialists to "plug-and-play" reasoning components. Technical Report DIST Technical Report 97-0049, University of Genova, Italy, 1997.

[BCGH98] P.G. Bertoli, J. Calmet, F. Giunchiglia, and K. Homann. Specification and combination of theorem provers and computer algebra systems. submitted for publication, 1998.

[Ber97] P. G. Bertoli. *Using OMRS in Practice: A Case Study with ACL2.* PhD thesis, University of Rome 3, 1997.

[CDE+98] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and J. Quesada. Maude: Specification and programming in rewriting logic, 1998. URL: `http://maude.csl.sri.com`.

[CGMT98] A. Coglio, F. Giunchiglia, J. Meseguer, and C. Talcott. Composing and controlling deduction in reasoning theories using mappings. Technical Report DIST and IRST Technical Report, IRST, Trento Italy and University of Genova, Italy, 1998. submitted for publication.

[CGPT97] A. Coglio, F. Giunchiglia, P. Pecchiari, and C. Talcott. A logic level specification of the nqthm simplification process. Technical report, IRST, University of Genova, Stanford University, 1997.

[Cla98] Manuel Clavel. *Reflection in General Logics, Rewriting Logic, and Maude.* PhD thesis, University of Navarre, 1998.

[CM96a] M. Clavel and J. Meseguer. Axiomatizing reflective logics and languages. In G. Kiczales, editor, *Reflection'96*, pages 263–288. Xerox PARC, 1996.

[CM96b] M. Clavel and J. Meseguer. Reflection in rewriting logic. In *Rewriting Logic Workshop'96*, number 4 in Electronic Notes in Theoretical Computer Science. Elsevier, 1996. URL: `http://www.elsevier.nl/locate/entcs/volume4.html`.

[CM97a] M. Cerioli and J. Meseguer. May I borrow your logic? (Transporting logical structure along maps). *Theoretical Computer Science*, 173:311–347, 1997.

[CM97b] M. Clavel and J. Meseguer. Internal strategies in a reflective logic. In *Proc. Workshop on Strategies in Automated Deduction, CADE-14*, 1997.

[Cog96] A. Coglio. The control component of OMRS. Master's thesis, University of Genova, Italy, 1996.

[DM98] F. Duran and J. Meseguer. Structured theories and institutions. In *Category Theory in Computer Science*, Electronic Notes in Theoretical Computer Science. Elsevier, 1998.

[DMT99] G. Denker, J. Meseguer, and C. Talcott. Rewriting Semantics of Distributed Meta Objects and Composable Communication Services, 1999. submitted.

[Dua99] C. H. C. Duarte. *Proof-theoretic Foundations for the Design of Extensible Software Systems.* PhD thesis, Imperial College, University of London, 1999.

[Dur99] Francisco Durán. *A Reflective Module Algebra with Applications to the Maude Language.* PhD thesis, University of Malaga, 1999.

[GPA96] F. Giunchiglia, P. Pecchiari, and A. Armando. Towards provably correct system synthesis and extension. *Future Generation Computer Systems*, 12(458):123–137, 1996.

[GPT94] F. Giunchiglia, P. Pecchiari, and C. L. Talcott. Reasoning theories: Towards an architecture for open mechanized reasoning systems. Technical Report 9409-15, IRST, November 1994. Also appears as Stanford University Computer Science Department Technical Note STAN-CS-94-TN-15.

[GPT96] F. Giunchiglia, P. Pecchiari, and C. Talcott. Reasoning theories: Towards an architecture for open mechanized reasoning systems. In *Workshop on Frontiers of Combining Systems FROCOS'96*, 1996.

[Mes89] J. Meseguer. General logics. In *Logic Colloquium 87*. North-Holland, 1989.

[Mes92] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[Mes98] José Meseguer. Research directions in rewriting logic. In U. Berger and H. Schwichtenberg, editors, *Computational Logic, NATO Advanced Study Institute, Marktoberdorf, Germany, July 29 – August 6, 1997*. Springer-Verlag, 1998.

[MOM97] Narciso Martí-Oliet and José Meseguer. Rewriting logic as a logical and semantic framework. In D. Gabbay, editor, *Handbook of Philosophical Logic*. Kluwer Academic Publishers, 1997.

[MT] J. Meseguer and C. Talcott. Reasoning theories and rewriting logic. (in preparation).

[MT98] J. Meseguer and C. Talcott. Mapping OMRS to Rewriting Logic. In C. Kirchner and H. Kirchner, editors, *2nd International Workshop on Rewriting Logic and its Applications, WRLA'98*, volume 15 of *Electronic Notes in Theoretical Computer Science*, 1998. URL: `http://www.elsevier.nl/locate/entcs/volume15.html`.

[MT99a] I. A. Mason and C. L. Talcott. Actor languages: Their syntax, semantics, translation, and equivalence. *Theoretical Computer Science*, 228(1), 1999.

[MT99b] J. Meseguer and C. Talcott. A Partial Order Event Model for Concurrent Objects. In *Proceedings of CONCUR'99: Concurency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 415–430, 1999.

[ST99] S. F. Smith and C. L. Talcott. Modular reasoning for actor specification diagrams. In P. Ciancariani, A. Fantechi, and R. Gorrieri, editors, *Formal Methods for Open Object-based Distributed Systems*, pages 313–330. Kluwer, 1999.

[Tal96a] C. L. Talcott. An actor rewriting theory. In J. Meseguer, editor, *Proc. 1st Intl. Workshop on Rewriting Logic and its Applications*, volume 4 of *Electronic Notes in Theoretical Computer Science*, pages 360–383. North Holland, 1996. `http://www1.elsevier.nl/mcs/tcs/pc/volume4.htm`.

[Tal96b] C. L. Talcott. Interaction semantics for components of distributed systems. In E. Najm and J-B. Stefani, editors, *1st IFIP Workshop on Formal Methods for Open Object-based Distributed Systems, FMOODS'96*, 1996. Proceedings published in 1997 by Chapman & Hall.

[Tal97] C. L. Talcott. Composable semantic models for actor theories. In M. Abadi and T. Ito, editors, *Theoretical Aspects of Computer Science*, number 1281 in Lecture Notes in Computer Science, pages 321–364. Springer-Verlag, 1997.

[Tal98] C. L. Talcott. Composable semantic models for actor theories. *Higher-Order and Symbolic Computation*, 11(3), 1998.

[Tal99] C. L. Talcott. Actor theories in rewriting logic, 1999. submitted for publication.

[VAT92] N. Venkatasubramanian, G. Agha, and C. L. Talcott. Scalable distributed garbage collection for systems of active objects. In *International Workshop on Memory Management, IWMM92, Saint-Malo*, LNCS, 1992.

[VAT99] N. Venkatasubramanian, G. Agha, and C. L. Talcott. Specifying composable services for QoS-based distributed resource management, 1999. submitted.

[Ven98] N. Venkatasubramanian. *Resource Management in Open Distributed Systems with Applications to Multimedia*. PhD thesis, University of Illinois, Urbana-Champaign, 1998.

[VT93] N. Venkatasubramanian and C. L. Talcott. A metaarchitecture for distributed resource management. In *Hawaii International Conference on System Sciences, HICSS-26*, January 1993.

[VT95] N. Venkatasubramanian and C. L. Talcott. Reasoning about Meta Level Activities in Open Distributed Systems. In *Principles of Distributed Computation (PODC '95)*, pages 144–153. ACM, 1995.