Open Mechanized Reasoning Systems: a Preliminary Report*

Alessandro Armando[1], Piergiorgio Bertoli[3], Alessandro Coglio[2]
Fausto Giunchiglia[3], Jose Meseguer[4], Silvio Ranise[1], and Carolyn Talcott[5]

[1] DIST, Università di Genova – Via all'Opera Pia 13, 16145 Genova, Italy
[2] Kestrel Institute – 3260 Hillview Avenue, Palo Alto, California 94304, U.S.A.
[3] IRST (Institute for Scientific and Technological Research) – 38050 Trento, Italy
[4] SRI International, 333 Ravenswood Av., Menlo Park, CA, USA
[5] Computer Science Department, Stanford University, CA 94305-2140, USA

Abstract. It is widely recognised that the integration of different (sub)-provers is a key issue in the construction of reasoning tools of practical usage. Unfortunately experience shows that effective integration is very difficult to achieve. The *Open Mechanized Reasoning Systems* (*OMRS*) Project started in 1992 with the objective to design a formal framework for the specification of state-of-the-art provers. The starting point of the OMRS approach is to structure the specification of a system in a logic component, a control component, and an interaction component. Crisply separating the concerns of the three layers, results in clearer and better specifications. The paper provides an informal and preliminary report of the OMRS Project and briefly illustrates the application of the OMRS specification framework to two challenge applications.

1 Introduction

It is widely recognised that the integration of different (sub-)provers is a key issue in the construction of reasoning tools of practical usage. This objective has been pursued by the automated reasoning community both *(i)* by embedding decision procedures inside general purpose reasoning systems (see, e.g., [17, 5, 18]) and—more recently—*(ii)* by interconnecting different reasoning tools (possibly implementing diverse reasoning paradigms) [12, 2, 6].

Experience shows that effective integration is very difficult to achieve. On the one hand, a main difficulty is that most of the research on decision procedures is focused on procedures delivering a 'yes-or-no' answer, whereas experience shows that in order to achieve effective integration much more sophisticated functionalities are required. On the other hand, the main problem with making existing systems interoperate is that most provers are packaged as stand-alone software with inadequately described interfaces. In both cases a fundamental problem is

* Thanks to Paolo Pecchiari for his contributions to the initial development of the OMRS Project. We are grateful to Enrico Giunchiglia for his encouragement and stimulating discussions.

the lack of a comprehensive conceptual framework for specifying the functionalities provided/required by state-of-the-art systems accounting for the wide variety of notions occurring in existing systems such as, e.g., consequence relations, annotations, control information, proof-strategies, incremental and restartable computation and deduction, interaction protocols, and communication facilities.

The *Open Mechanized Reasoning Systems* (*OMRS*) Project started in 1992 with the objective to design a formal framework for the specification of state-of-the-art provers. The starting point of the OMRS approach is to structure the specification of a system in a logic component, a control component, and an interaction component, thereby suggesting the following equation:

$$\text{OMRS} = \text{LOGIC} + \text{CONTROL} + \text{INTERACTION}$$

Preliminary but significant results have been obtained in the application of the OMRS framework for supporting *(i)* the definition and the development of provers as open architectures usable in a "plug-and-play" fashion, and *(ii)* the design and development of proof-checkable and customizable reasoning systems. We are also thinking of the formal synthesis from OMRS specifications of provably correct, efficient, and re-usable reasoning systems.

This paper is intended to be a preliminary and informal report of the OMRS Project. Space limitations prevent us from giving the details, but references to the relevant publications are given. (A comprehensive and up to date description as well as a list of the publications of the OMRS Project are available at [19].)

The paper is organized as follows. Section 2 provides the key ideas of the OMRS specification framework. Section 3 illustrates how the OMRS framework has been used to guide the re-engineering of ACL2 [13] into a proof-checkable and customizable reasoning tool. Section 4 illustrates a methodology which allows the extraction and lifting of reasoning specialists integrated in existing systems into reusable and implementation independent reasoning components to be used in a "plug-and-play" fashion. Some final remarks are given in Section 5.

2 The OMRS Specification Framework

Starting from the consideration that any reasoning system, as such, performs deductions within some logic(s), guided by some (more or less complex) heuristics, and exhibits some interaction capabilities, an OMRS specification of a reasoning system is structured in a logic component, a control component, and an interaction component. The logic component provides a description of the assertions manipulated by the system and the elementary deductions upon them; the control component allows for the specification of the strategies guiding the construction of complex deductions out of the elementary ones; finally the interaction component specifies how the system interacts with the external world (including human users and other provers). Crisply separating the concerns of the three layers, results in clearer and better specifications. This is an important issue as it allows us to cope with the complexity of existing systems.

The logic component of an OMRS specification consists of a *reasoning theory* (*RTh*) [11], containing a set of assertions (called *sequents*), a set of inference *rules*, and possibly a set of *constraints* used for specifying applicability conditions of the rules. Constraint manipulation is specified by means of another RTh (whose sequents are constraints) which is said to be *nested* in the first one. Nesting can take place up to an arbitrary level. In order to support schematic and provisional reasoning, sequents and constraints of an RTh can be schematic (i.e., contain place-holders for unspecified pieces of syntax). Deductions are expressed by labeled graphs, called *reasoning structures*, which generalize the notion of proof tree by allowing representation of proof fragments, sharing of deductions, explicit treatment of constraints and inferences on them, and arbitrary nesting of reasoning structures.

As most successful provers are built by integrating various specialized modules (e.g., a rewriter and a decider for linear arithmetic), our framework provides a *composition* operation over reasoning theories [10]. The RTh for the whole system is obtained by composing those for the constituent modules. Our composition operation allows RThs to share some language, and guarantees that the shared language and its meaning are preserved by the composition.

For the control component of an OMRS specification, it is necessary to specify the non-logical (i.e., control) data structures manipulated by the systems (e.g., clause indexes, the set of procedures which have produced a certain sequent). We do that by augmenting sequents with *annotations* encoding control information, and lifting rules to deal with annotations (e.g., update of clause indexes). More precisely, the control component includes an RTh, called *annotated RTh*, together with an *erasing mapping* specifying how the annotated RTh can be mapped into an RTh embodying the logic content of the annotated RTh. This approach gives two major advantages. One is that the manipulation of heuristic information is specified in a declarative way, and therefore clearly it shows (via the erasing mapping) the underlying logical deduction. The other is that composition of RThs lifts from the logic to the control component: the composition of annotated RThs specifies how the various sub-systems share data structures carrying both logical and control information.

Work is in progress to define a formalism to specify strategies of application of annotated rules and the construction of the corresponding (annotated) reasoning structures. Some preliminary work has been done in [9], where a tactical language is employed to build tree-shaped reasoning structures.

Concerning the interaction component of an OMRS specification, this is mostly work in progress. In any case, this component should specify primitives through which a system can have a bidirectional interaction with the external world, and should specify how primitives result in the activation of deduction strategies specified in the control component. For instance, some systems support queries to add an axiom to their internal database, to prove a theorem, and so on.

There is a strong connection between OMRS and rewriting logic [15, 14]. In particular there is a close correspondence between the notions of rewrite theory

in rewriting logic and reasoning theory in the OMRS framework. Work is in progress to make this connection precise [16]. A substantial benefit of this work is that it will make tools for rewriting logic applicable to OMRS. Specifically the Maude language [7] can serve as a formal notation for reasoning theories. Using the Maude tool this allows easy prototyping of systems described by reasoning theories. The reflective capability of Maude [8] provides a means of defining strategy languages, for describing and executing OMRS strategies, and for constructing and manipulating reasoning structures. Mappings between reasoning theories and composition of reasoning theories can also be defined and executed in Maude.

3 Building Flexible and Proof-Checkable Reasoning Tools

The OMRS specification framework has been used to guide the redesign of the ACL2 system [13] into a flexible and proof-checkable reasoning tool. The motivation for the case study (whose details can be found in [3]) is that, although ACL2 is known as one of the most powerful and reliable provers available, its proofs do not result into structured proof objects; thus, it is impossible to formally analyze (e.g., check) its results as required in many critical applications. Furthermore, we want the system to be easily customizable. Finally, by redesigning a state-of-the-art prover through OMRS, we aimed at verifying both the expressivity and the practical usability of the OMRS formalism.

The case study is focused on the top-level inference engine of the prover, i.e. the *waterfall*[1], but the fragment considered is complex enough to suggest that the methodology can scale up to the redesign of the whole system. Following the OMRS approach the data structures manipulated by the waterfall have been turned into annotated sequents, and the inference processes have been modeled as tactics working on annotated sequents and implementing rules of a suitably defined annotated reasoning theory. Tactics are combined by means of a tactical language similar in spirit to LCF's, but suitably extended so to deal with the reuse of sub-deductions. This required the design of an interpreter for the tactical language, which has been embedded within the ACL2 system.

From a methodological standpoint, the application of OMRS has required a deep study of the structures and the processes invoked by the waterfall in order to enforce a crisp distinction between the logic, the control and the interaction aspects, which are deeply intertwined in the original code. We have been able to reuse most of the original data structures, by providing mapping functions to achieve the desired splitting between the aforementioned aspects. The approach has been applied uniformly, starting from the OMRS rewriting of the ACL2 processes as primitive annotated tactics, up to the compound tactical expression representing the whole waterfall.

As a result of our effort, a partially OMRS-redesigned version of the ACL2 prover has been implemented. Although a prototype, it constitutes a first experi-

[1] The waterfall can be described as a recursive application of a chain of backward heuristics called "processes".

mental evidence of the feasibility of adopting the OMRS framework to (re-)design complex systems, and provides some interesting indications regarding to which degree OMRS achieves its aims. The prototype is capable to present the results of a successful deduction by means of a reasoning structure. Several degrees of flexibility in the presentation are possible; e.g., it is possible to choose the granularity of the presentation, or to hide the heuristic details of the proof, showing only the logically relevant steps. The reasoning structure can be conveniently displayed by means of a graph editor, and is amenable to formal proof checking. The prototype allows the user to customize the system by creating new primitive and compound tactics, and by reusing the waterfall tactics.

4 "Plug-and-Play" Reasoning Components

The OMRS specification framework has been applied to define a methodology for turning reasoning specialists embedded into existing systems into reusable and implementation independent reasoning components (i.e. open architectures capable of exchanging selected sets of logical services with the environment) to be used in a "plug-and-play" fashion. This is an important objective for two reasons. First, the development of a new reasoning specialist is a very difficult and time consuming activity. Second, existing reasoning systems represent a real cornucopia of powerful reasoning specialists. The methodology has been tested by applying it to a challenge case study: the extraction of the linear arithmetic procedure incorporated in NQTHM [4]. (A detailed description of both the methodology and the case study can be found in [1].)

The first step of the methodology amounts to modeling an existing reasoning system as a set of reasoning specialists glued together by means of an integration schema (i.e. a specification of the interplay between the reasoning specialists). In our case study, this activity required a careful analysis of both the actual implementation code and the report [5]. The OMRS specification framework provided us with the conceptual background to carry out a rational reconstruction of the integration schema employed by Boyer and Moore. The second step of the methodology amounts to lifting the reasoning specialists and the integration schema into a set of reasoning components and an interaction schema, respectively. A reasoning component is composed by a set of logical services together with an interaction protocol. The interaction protocol governs the exchange of the logical services between a reasoning component and the environment. In our case study, this amounted to providing the linear arithmetic specialist with interaction capabilities and an interaction protocol. Finally, we glued the reasoning component back with the rest of the system, thereby obtaining a combined system as opposed to the (original) integrated one. A comparative experimental analysis between the combined and the original system confirms the viability of the approach of lifting embedded reasoning specialists to "plug-and-play" reasoning components. We are currently planning to interface the linear arithmetic component we extracted from NQTHM with other provers.

The case study we chose is of particular interest because of the following three reasons. The first two stem from features shared by most of the decision procedures embedded in state-of-the-art provers: *incrementality* and *restartability*. Incrementality is used for efficiency reasons. Restartability is necessary to support contextual rewriting. In fact the context stored into the linear arithmetic procedure must be changed as soon as the focus of rewriting changes. The third reason is that the interaction between the prover and the procedure is particularly sophisticated. In fact when the linear arithmetic procedure is asked to establish whether a given linear facts is entailed by the context, it first tries to prove this fact by linear arithmetic reasoning only, and—upon failure—it asks back the prover for facts involving certain user-defined functions occurring in the context. This results in a particularly complex form of interaction whereby the prover and the linear arithmetic specialist can engage in an arbitrarily long sequence of requests and replies.

5 Conclusions

The OMRS Project provides a specification framework for the construction of reasoning systems which are open in the sense that they can be easily (or perhaps just more easily than current technology allows) integrated together. The application of the framework to the case studies described in this paper give evidence of the viability of the approach.

References

[1] A. Armando and S. Ranise. From Integrated Reasoning Specialists to "Plug-and-Play" Reasoning Components. In *Fourth International Conference on Artificial Intelligence and Symbolic Computation (AISC98). Plattsburgh, NY, USA. September 16-18*, 1998.

[2] C. Ballarin, K. Homann, and J. Calmet. Theorems and Algorithms: An Interface between Isabelle and Maple. In *ISAAC'95*, pages 150–157, Canada, 1995.

[3] P.G. Bertoli. *Using OMRS in practice: a case study with* ACL2. PhD thesis, Computer Science Dept., University Rome 3, Rome, 1997. Forthcoming.

[4] R.S. Boyer and J.S. Moore. *A Computational Logic.* Academic Press, 1979. ACM monograph series.

[5] R.S. Boyer and J.S. Moore. Integrating Decision Procedures into Heuristic Theorem Provers: A Case Study of Linear Arithmetic. *Machine Intelligence*, 11:83–124, 1988.

[6] William Chan, Richard J. Anderson, Paul Beame, and David Notkin. Combining constraint solving and symbolic model checking for a class of systems with non-linear constraints. In Orna Grumberg, editor, *Computer Aided Verification, 9th International Conference, CAV'97 Proceedings*, volume 1254 of *Lecture Notes in Computer Science*, pages 316–327, Haifa, Israel, June 1997. Springer-Verlag.

[7] M. Clavel, S. Eker, P. Lincoln, and J. Meseguer. Principles of maude. In *Rewriting Logic Workshop'96*, 1996.

[8] M. Clavel and J. Meseguer. Reflection in rewriting logic. In *Rewriting Logic Workshop'96*, 1996.

[9] A. Coglio. "Definizione di un formalismo per la specifica delle strategie di inferenza dei sistemi di ragionamento meccanizzato e sua applicazione ad un sistema allo stato dell'arte", 1996. Master thesis, DIST - University of Genoa (Italy).

[10] Alessandro Coglio, Fausto Giunchiglia, José Meseguer, and Carolyn L. Talcott. Composing and controlling search in reasoning theories using mappings. 1998. Submitted to the Second International Workshop on 'Frontiers of Combining Systems' (FroCoS'98).

[11] F. Giunchiglia, P. Pecchiari, and C. Talcott. Reasoning Theories: Towards an Architecture for Open Mechanized Reasoning Systems. Technical Report 9409-15, IRST, Trento, Italy, 1994. Also published as Stanford Computer Science Department Technical note number STAN-CS-TN-94-15, Stanford University. Short version published in Proc. of the First International Workshop on Frontiers of Combining Systems (FroCoS'96), Munich, Germany, March 1996.

[12] J. Harrison and L. Théry. A Sceptic's Approach to Combining HOL and Maple. To appear in the JAR, 1997.

[13] M. Kaufmann and J. S. Moore. ACL2 Version 1.8 User's Manual. Available on line at `http://www.cs.utexas.edu/users/moore/acl2/index.html`.

[14] 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.

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

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

[17] C. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *TOPLAS*, 1(2):245–257, 1979.

[18] Robert E. Shostak. Deciding combinations of theories. *Journal of the ACM*, 31(1):1–12, January 1984.

[19] The OMRS Taskforce. The Open Mechanized Reasoning Systems Project WWW Page. `http://www.mrg.dist.unige.it/omrs/`.