Parameterized strategies specification in Maude*

Narciso Martí-Oliet, Isabel Pita, Rubén Rubio, Alberto Verdejo

Universidad Complutense de Madrid, Spain
{narciso,ipandreu,rubenrub,jalberto}@ucm.es

Strategies are ubiquitous in Computer Science. As recipes to tackle search problems and bound nondeterminism, they appear in algorithms, automatic deduction, language semantics, artificial intelligence, ... Some of these examples can be specified and analyzed compositionally, abstracting not only data representation and rules but also their control. A convenient way of expressing parametric control is parameterized strategies.

Maude [2] is a declarative high-level language based on *rewriting logic* [8], allowing the description, execution and analysis of many models of concurrent and distributed systems at different levels. In terms of *membership equational logic* [1], we can specify sorts, symbols, equations and membership axioms. Then we can add rewrite rules to represent transitions of a concurrent system, which need neither be deterministic, nor confluent, nor terminating. Above the previous specification, one level up, we can control how rules are applied using a strategy language [5, 6], which has been completed recently.

The basic component of the Maude strategy language is rule application. Rules are identified by labels, substitutions can be given to instantiate their variables before application, and additional strategies can be provided to guide the rule's rewriting conditions (if any). A family of `match` operators allows testing whether the term being rewritten matches a given pattern and satisfies equational conditions. Composed strategies are built through union, concatenation, regular expressions like $E*$ and $E+$, and conditionals. Another operator applies strategies to specific subterms, and finally, strategy modules allow defining named strategies with recursion. These strategy modules can take parameters (like the traditional modules in Maude) where required sorts, symbols, and strategies are declared using a *theory*. Any module can be said to comply with a theory by means of a *view*, which is later used to instantiate parameterized modules.

The strategy language has already been exploited in the specification of algorithms, inference systems, and language semantics. Milner's CCS, ambient calculus, and the semantics of the parallel functional language Eden are addressed in [7], and in [9] strategies deal with completion procedures. These examples are likely to be expressed and generalized using control parameterization with strategies, whose implementation was not available at that time. Once expressed in this way, the specified systems can be both executed and tested with different alternative strategies provided as parameters, or analyzed at different levels with specific tools, like the LTL model checker.

Generic algorithmic schemes and skeletons [4] are other good candidates for being expressed in those terms. *Backtracking* is a simple example. A backtracking problem is defined by a theory which contains a `State` sort, two equational predicates `isOk`

* Research partially supported by MINECO Spanish project *TRACES* (TIN2015-67522-C3-3-R), and Comunidad de Madrid project N-Greens Software-CM (S2013/ICE-2731).

and `isSolution` to test whether a given state is valid or solution respectively, and the strategy `expand` to generate the next search states. A parameterized strategy module defines backtracking as another strategy which keeps expanding the valid states until it finds a solution:

```
sd solve := (match S s.t. isSolution(S)) ? idle :
                 (expand ; match S s.t. isOk(S) ; solve)
```

Then a particular example like the labyrinth escape problem can be expressed instantiating `State` with 2D positions, `isOk` to a function that checks that the position is inside the labyrinth bounds, and `isSolution` to be true at the exits. `expand` is then a union of rules for moving from the current position to its neighbors.

As a extension of Maude 2.7.1 [3], the strategy language implementation has been recently completed in C++. Now, we are analysing the previous examples and writing new ones to take full advantage of parameterization, including the λ-calculus and a simple functional language with several reduction strategies, the simplex algorithm with different pivoting rules, and ecological models.

References

[1] Adel Bouhoula, Jean-Pierre Jouannaud, and José Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236(1):35–132, 2000.

[2] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *All About Maude-A High-Performance Logical Framework*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.

[3] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. *Maude Manual (v 2.7.1)*. July 2016.

[4] J. Darlington, A. J. Field, P. G. Harrison, P. H. J. Kelly, D. W. N. Sharp, Q. Wu, and R. L. While. Parallel programming using skeleton functions. In Arndt Bode, Mike Reeve, and Gottfried Wolf, editors, *PARLE '93 Parallel Architectures and Languages Europe*, pages 146–160. Springer, 1993.

[5] Steven Eker, Narciso Martí-Oliet, José Meseguer, and Alberto Verdejo. Deduction, strategies, and rewriting. *Electron. Notes Theor. Comput. Sci.*, 174(11):3–25, July 2007.

[6] Narciso Martí-Oliet, José Meseguer, and Alberto Verdejo. Towards a strategy language for maude. *Electron. Notes Theor. Comput. Sci.*, 117:417–441, 2005. Proceedings of the Fifth International Workshop on Rewriting Logic and Its Applications (WRLA 2004).

[7] Narciso Martí-Oliet, Miguel Palomino, and Alberto Verdejo. Strategies and simulations in a semantic framework. *Journal of Algorithms*, 62(3):95–116, 2007.

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

[9] Alberto Verdejo and Narciso Martí-Oliet. Basic completion strategies as another application of the maude strategy language. In Santiago Escobar, editor, *Proceedings 10th International Workshop on Reduction Strategies in Rewriting and Programming, WRS 2011*, pages 17–36, 2011.