The Effects of Modalities in Separation Logics (Extended Abstract)

Stéphane Demri [1]

New York University, USA & CNRS, France

Morgan Deters [2]

New York University, USA

Abstract

Like modal logic, temporal logic, or description logic, separation logic has become a popular class of logical formalisms in computer science, conceived as assertion languages for Hoare-style proof systems with the goal to perform automatic program analysis. We present similarities with modal and temporal logics, and we present landmark results about decidability, complexity and expressive power.

Keywords: Separation logic, decidability, computational complexity, expressive power, temporal logic, modal logic, first-order logic, second-order logic

When separation logic joins the club Introducing new logics is always an uncertain enterprise since there must be sufficient interest to use new formalisms. In spite of this hurdle, we know several recent success stories. For instance, even though a pioneering work on symbolic modal logic by Lewis appeared in 1918 [20], the first monographs on symbolic modal logic appear some fifty years later, see e.g. [16]. Nowadays, modal logic is divided into many distinct branches and remains one of the most active research fields in logic and computer science. Additionally, the introduction of temporal logic to computer science, due to Pnueli [25], has been a major step in the development of model-checking techniques, see e.g. [10,3]. This is now a well-established approach for the formal verification of computer systems: one models the system to be verified by a mathematical structure (typically a directed graph) and expresses behavioral properties in a logical formalism (typically a temporal logic). Verification by model-checking [10] consists of developing algorithms whose goal is

[1] demri@cs.nyu.edu. Work partially supported by the EU Seventh Framework Programme under grant agreement No. PIOF-GA-2011-301166 (DATAVERIF).

[2] mdeters@cs.nyu.edu. Work partially supported by the Air Force Office of Scientific Research under award FA9550-09-1-0596, and the National Science Foundation under grant 0644299.

to verify whether the logical properties are satisfied by the abstract model. The development of tools is done in parallel with the design of techniques to optimize the verification process. Apart from the development of model-checkers such as Cadence SMV, SPIN, or Uppaal, the transfer towards industrial applications is also present in research and development units. The development of description logics for knowledge representation has also followed a successful path, thanks to a permanent interaction between theoretical works, pushing ever further the high complexity and undecidability borders, and more applied works dedicated to the design of new tools and the production of more and more applications, especially in the realm of ontology languages. The wealth of research on description logic is best illustrated by [2], in which can be found many chapters on theory, implementations, and applications. By contrast, Chapter 1 of [2] provides a gentle introduction to description logics and recalls that its roots can be traced back a few decades. It is well-known that modal logic, temporal logic, and description logic have many similarities even though each family has its own research agenda. For instance, models can be (finite or infinite) graphs, the classes of models range from concrete ones to more abstract ones, and any above-mentioned class includes a wide range of logics and fragments. In this work, we deal with another class of logics, separation logic, that has been introduced quite recently (see e.g. [17,26]). Separation logic is the subject of tremendous interest, leading to many works on theory, tools and applications (mainly for the automatic program analysis). Any resemblance to modal, temporal, or description logic is certainly not purely coincidental—but separation logic also has its own assets.

In the possible-world semantics for modal logic, the modal operator $\Box$ [resp. $\Diamond$] corresponds to universal [resp. existential] quantification on successor worlds, and these are essential properties to be stated, partly explaining the impact of Kripke's discovery [18,12]. Similarly, the ability to divide a model into two disjoint parts happens to be a very natural property. This might explain the success of separation logic: disjoint memory states can be considered, providing an elegant means to perform local reasoning. *Separation* is a key concept that has been already introduced in interval temporal logic (ITL) [23] with the *chop* operator (and probably in many other logical formalisms such as in graph logics [13]) and therefore, the development of separation logic can be partly explained by the relevance of the separation concept. Its impressive development can be also justified by the fact that separation logic extends Hoare logic for reasoning about programs with dynamic data structures, meeting also industrial needs as witnessed by the recent acquisition of Monoidics Ltd. by Facebook.

Separation and composition Separation logic has been introduced as an extension of Hoare logic [15] to verify programs with mutable data structures [17,26]. A major feature is the ability to reason locally in a modular way, which can be performed thanks to the separating conjunction $*$ that allows one to state properties in disjoint parts of the memory. Moreover, the adjunct implication $-\!\!*$ asserts that whenever a fresh heap satisfies a property,

its composition with the current heap satisfies another property. This is particularly useful when a piece of code mutates memory locally, and we want to state some property of the entire memory (such as the preservation of data structure invariants). In a sense, if modal logic is made for reasoning about necessity and possibility, separation logic is made for reasoning about separation and composition. Of course this type of statement is an oversimplification—apart from the fact that it may appear a bit old-fashioned to most modal logicians—but this may help to get a first picture. As a taste of separation logic, it is worth observing that models can be finite graphs and the classes of models range from concrete ones (with heaps for instance) to very abstract ones (e.g., cancellative partial commutative monoids). While evaluating a formula, models can be updated as in public announcement logics.

Smallfoot was the first implementation to use separation logic, its goal to verify the extent to which proofs and specifications made by hand could be treated automatically [4]. The automatic part is related to assertion checking, but the user has to provide preconditions, postconditions, and loop invariants. A major step has been then to show that the method is indeed scalable [28]. In a sense, the legitimate question about the practical utility of separation logic was quickly answered, leading to a new generation of tools including Slayer developed by Microsoft Research, Space Invader [14,28], and Infer [7]. Actually, nowadays, many tools support separation logic as an assertion language and, more importantly, in order to produce interactive proofs with separation logic, several proof assistants encode the logic, see e.g. [27].

From the very beginning, the theory of separation logic has been an important research thread even if not always related to automatic verification. This is not very surprising since separation logic can be understood as a concretization of bunched logic BI which is a general logic of resources with a nice proof theory [24]. Besides, as for modal and temporal logics, the relationships between separation logic, and first-order or second-order logics have been the source of many characterizations and works. This is particularly true since the separation connectives are second-order in nature, see e.g. [21,19,8,5]. For instance, separation logic is equivalent to a Boolean propositional logic [22,21] if first-order quantifiers are disabled. Similarly, the complexity of satisfiability and model-checking problems for fragments of separation logic has been examined [9,26,11,1,6]. In [9], the model-checking and satisfiability problems for propositional separation logic are shown PSPACE-complete; this is done by proving a small model property.

Content The goal of this work is twofold. First, we would like to emphasize the similarities between separation logic and modal and temporal logics. Our intention is to pinpoint the common features in terms of models, proof techniques, motivations, and so forth. Second, we wish to present landmark results about decidability, complexity, and expressive power, providing a survey on the theoretical side of separation logic. These are standard themes for studying logics in computer science, and we deliberately focus on the logical aspects of separation logic.

References

[1] Antonopoulos, T., N. Gorogiannis, C. Haase, M. Kanovich and J. Ouaknine, *Foundations for decision problems in separation logic with general inductive predicates*, in: *FOSSACS'14*, Lecture Notes in Computer Science **8412** (2014), pp. 411–425.

[2] Baader, F., D. Calvanese, D. McGuinness, D. Nardi and P. Patel-Schneider, editors, "The Description Logic Handbook: Theory, Implementation and Applications," Cambridge University Press, 2003.

[3] Bérard, B., M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci and P. Schnoebelen, "Systems and Software Verification, Model-Checking Techniques and Tools," Springer, 2001.

[4] Berdine, J., C. Calcagno and P. O'Hearn, *Smallfoot: Modular automatic assertion checking with separation logic*, in: *FMCO'05*, Lecture Notes in Computer Science **4111** (2005), pp. 115–137.

[5] Brochenin, R., S. Demri and E. Lozes, *On the almighty wand*, Information and Computation **211** (2012), pp. 106–137.

[6] Brotherston, J., C. Fuhs, N. Gorogiannis and J. Navarro Perez, *A decision procedure for satisfiability in separation logic with inductive predicates*, in: *LICS'14*, 2014, to appear.

[7] Calcagno, C. and D. Distefano, *Infer: An automatic program verifier for memory safety of C programs*, in: *NASA Formal Methods*, Lecture Notes in Computer Science **6617** (2011), pp. 459–465.

[8] Calcagno, C., P. Gardner and M. Hague, *From separation logic to first-order logic*, in: *FOSSACS'05*, Lecture Notes in Computer Science **3441** (2005), pp. 395–409.

[9] Calcagno, C., P. O'Hearn and H. Yang, *Computability and complexity results for a spatial assertion language for data structures*, in: *FSTTCS'01*, Lecture Notes in Computer Science **2245** (2001), pp. 108–119.

[10] Clarke, E., O. Grumberg and D. Peled, "Model checking," The MIT Press Books, 2000.

[11] Cook, B., C. Haase, J. Ouaknine, M. Parkinson and J. Worrell, *Tractable reasoning in a fragment of separation logic*, in: *CONCUR'11*, Lecture Notes in Computer Science **6901** (2011), pp. 235–249.

[12] Copeland, J., *The genesis of possible worlds semantics*, Journal of Philosophical Logic **31** (2002), pp. 99–137.

[13] Dawar, A., P. Gardner and G. Ghelli, *Expressiveness and complexity of graph logic*, Information and Computation **205** (2007), pp. 263–310.

[14] Distefano, D., P. O'Hearn and H. Yang, *A local shape analysis based on separation logic*, in: *TACAS'06*, Lecture Notes in Computer Science **3920** (2006), pp. 287–302.

[15] Hoare, C. A. R., *An axiomatic basis for computer programming*, Communications of the ACM **12** (1969), pp. 576–580.

[16] Hughes, G. and M. Cresswell, "An introduction to modal logic," Methuen and Co., 1968.

[17] Ishtiaq, S. and P. O'Hearn, *BI as an assertion language for mutable data structures*, in: *POPL'01* (2001), pp. 14–26.

[18] Kripke, S., *A completeness theorem in modal logic*, The Journal of Symbolic Logic **24** (1959), pp. 1–14.

[19] Kuncak, V. and M. Rinard, *On spatial conjunction as second-order logic*, Technical Report MIT–CSAIL–TR–2004–067, MIT CSAIL (2004).

[20] Lewis, C. I., "A survey of symbolic logic," University of California Press, Berkeley, 1918.

[21] Lozes, E., "Expressivité des Logiques Spatiales," Phd thesis, ENS Lyon (2004).

[22] Lozes, E., *Separation logic preserves the expressive power of classical logic*, in: *SPACE'04*, 2004.

[23] Moszkowski, B., *Reasoning about digital circuits*, Technical Report STAN-CS-83-970, Dept. of Computer Science, Stanford University, Stanford, CA (1983).

[24] O'Hearn, P. and D. Pym, *The logic of bunched implications*, Bulletin of Symbolic Logic **5** (1999), pp. 215–244.

[25] Pnueli, A., *The temporal logic of programs*, in: *FOCS'77* (1977), pp. 46–57.

[26] Reynolds, J., *Separation logic: a logic for shared mutable data structures*, in: *LICS'02* (2002), pp. 55–74.

[27] Tuerk, T., "A separation logic fragment for HOL," Ph.D. thesis, University of Cambridge (2011).
[28] Yang, H., O. Lee, J. Berdine, C. Calcagno, B. Cook, D. Distefano and P. O'Hearn, *Scalable shape analysis for systems code*, in: *CAV'08*, Lecture Notes in Computer Science **5123** (2008), pp. 385–398.