# Symbolic Equivalences for Open Systems

**(Extended Abstract)**★

Paolo Baldan[1], Andrea Bracciali[2], and Roberto Bruni[2,3]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italia.
[2] Dipartimento di Informatica, Università di Pisa, Italia.
[3] CS Department, University of Illinois at Urbana-Champaign, USA.
`baldan@dsi.unive.it, braccia@di.unipi.it, bruni@di.unipi.it`

The widespread diffusion of mobile devices and web applications calls for linguistic and meta-theoretical frameworks suited to reason about open systems, i.e., autonomous software modules where mobile software components can be dynamically plugged in to carry on their computations. In the literature, process calculi (PC) have been often proposed as a paradigm for the specification and analysis of such open systems. Not only they facilitate rigorous system analysis by focusing on the aspects of interest (e.g. security, distribution, mobility), but they can also offer the basis for prototypical implementation and for verification tools. For example, many running implementations exist of languages based on calculi that were originally proposed as benchmarks to experiment and study few basic primitives (e.g. name passing in $\pi$-calculus [16], process passing in HO$\pi$-calculus [18], cryptographic messages in spi-calculus [1], distributed dataspaces in KLAIM [10], mobile environments in ambient-calculus [8]).

However, PC techniques are mostly devised for the study of *components* (i.e. closed terms of the calculus) rather than *coordinators* (i.e. contexts with holes marked by process variables). In particular, the operational semantics and several semantic equivalences (e.g., based on either bisimulation or traces or testing) are often defined directly over components, while their extensions to coordinators usually require an additional effort. Roughly, an equivalence $\approx$ defined on components can be lifted to coordinators by letting $C[X_1,...,X_n] \approx_{\mathrm{univ}} D[X_1,...,X_n]$ when $C[p_1,...,p_n] \approx D[p_1,...,p_n]$ for all components $p_1,...,p_n$. In the case of bisimilarity, this means that the coalgebraic techniques applicable to components fall short for coordinators, since the definition involves universal quantification on components. In general, efficient finite-state verification techniques designed for $\approx$ can hardly be extended to $\approx_{\mathrm{univ}}$, whose definition represents a source of infinity. To overcome this problem, the idea is to resort to symbolic techniques—e.g. in the style of [13,19]—that encompass universal quantification, easing the analysis and verification of coordinators' properties.

Our proposal [2,3] relies on *symbolic transition systems* (STSs), i.e., transition systems where states are coordinators and transitions are labelled by logic formulae:

$$C[X_1,...,X_n] \xrightarrow{\varphi_1,...,\varphi_n} D[Y_1,\ldots,Y_m]$$

meaning that the step can be performed by $C[p_1,...,p_n]$ whenever each process $p_i$ satisfies the corresponding formula $\varphi_i$. The logic where the formulae $\varphi_i$'s live and the notion

★ Research supported by the MIUR Project COFIN 2001013518 CoMeta and by the FET-GC Projects IST-2001-32747 Agile, IST-2001-32617 Myths and IST-2001-32530 Socs. The third author is also partially supported by an Italian CNR fellowship.

of satisfaction $\models$ must be of course targeted to the PC under study. In general, the logic may involve both spatial and temporal aspects of components, in the style of the *spatial logic* of [7,9].

Once an STS is available, a hierarchy of equivalences can be defined directly over coordinators STS, including bisimilarity and trace equivalences. Let $\approx_{sts}$ denote one of such equivalences, defined over the STS, and let $\approx_{\mathrm{univ}}$ be be the standard extension to coordinators (by universal quantification) of the corresponding equivalence over components. Whenever the STS satisfies suitable properties (a sort of correctness and completeness w.r.t. the transition system over components) the equivalence $\approx_{sts}$ can be proved to be approximations $\approx$, i.e., $\approx_{sts}$ is finer than $\approx$.

Quite importantly, the STSs of interest can be automatically derived from an SOS specification of the transition system over components whenever the SOS are in rather general formats (e.g., *algebraic format* [12,5] or *linear positive gsos format*). This can be achieved via a very simple Prolog program, where the SOS rules are translated to Horn clauses in the obvious way and an additional rule handles behavioural constraints (exploiting the cut operator).

*Related work.* The papers [21,15,20] face the somehow dual problem of synthesizing, from a given reduction system, a labelled transition system where bisimilarity is a congruence. They have motivated and inspired our quest for a set of labels powerful enough to model the maximal classes of components with whom coordinators can react. The work on spatial logics [9,7] have suggested us an elegant mathematical tool for expressing both structural and temporal constraints in the labels. It is worth mentioning that spatial formulae share many analogies with the topological modalities introduced separately in [11] when proposing a verification logic for rewriting logic.

A symbolic approach to bisimulation in the case of value passing calculi, where actions are parametrised over a possibly infinite set, has been explored in [13].

Among other frameworks where the semantics of components and coordinators is defined uniformly, let us mention *tile logic* (TL) [12,4], *conditional transition systems* (CTS) [17] and *context systems* (CS) [14], which come also equipped with different formats for guaranteeing that bisimilarity is a congruence. While models based on TL, CTS and CS can be easily translated in our framework, the use of spatial formulae makes our approach applicable to a wider class of calculi.

The idea of using unification for building formulae comes from Logic Programming and more precisely by its view as an interactive system presented in [6].

## References

1. M. Abadi and A.D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Inform. and Comput.*, 148(1):1–70, 1999.
2. P. Baldan, A. Bracciali, and R. Bruni. Bisimulation by unification. In H. Kirchner and Ringeissen, editors, *Proceedings of AMAST 2002, 9th International Conference on Algebraic Methodology And Software Technology*, volume 2422 of *Lect. Notes in Comput. Sci.*, pages 254–270, 2002.
3. P. Baldan, A. Bracciali, and R. Bruni. A theory of symbolic equivalences. Submitted., 2002.

4. R. Bruni, D. de Frutos-Escrig, N. Martí-Oliet, and U. Montanari. Bisimilarity congruences for open terms and term graphs via tile logic. In C. Palamidessi, editor, *Proceedings of CONCUR 2000, 11th International Conference on Concurrency Theory*, volume 1877 of *Lect. Notes in Comput. Sci.*, pages 259–274. Springer Verlag, 2000.
5. R. Bruni, J. Meseguer, and U. Montanari. Executable tile specifications for process calculi. In J.-P. Finance, editor, *Proceedings of FASE'99, Fundamental Approaches to Software Engineering*, volume 1577 of *Lect. Notes in Comput. Sci.*, pages 60–76. Springer Verlag, 1999.
6. R. Bruni, U. Montanari, and F. Rossi. An interactive semantics of logic programming. *Theory and Practice of Logic Programming.*, 1(6):647–690, 2001.
7. L. Caires. *A Model for Declarative Programming and Specification with Concurrency and Mobility*. PhD thesis, Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, 1999.
8. L. Cardelli and A.D. Gordon. Mobile ambients. In M. Nivat, editor, *Proceedings of FoSSaCS'98, Foundations of Software Science and Computational Structures*, volume 1378 of *Lect. Notes in Comput. Sci.*, pages 140–155. Springer Verlag, 1998.
9. L. Cardelli and A.D. Gordon. Anytime, anywhere. modal logics for mobile ambients. In *Proceedings of 27th ACM Symposium on Principles of Programming Languages*, pages 365–377. ACM, 2000.
10. R. De Nicola, G. Ferrari, and R. Pugliese. KLAIM: a kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering*, 24(5):315–330, 1998.
11. J.L. Fiadeiro, T. Maibaum, N. Martí-Oliet, J. Meseguer, and I. Pita. Towards a verification logic for rewriting logic. In D. Bert, C. Choppy, and P. Mosses, editors, *Proceedings of WADT'99, 14th International Workshop on Recent Trends in Algebraic Development Techniques*, volume 1827 of *Lect. Notes in Comput. Sci.*, pages 438–458. Springer Verlag, 2000.
12. F. Gadducci and U. Montanari. The tile model. In G. Plotkin, C. Stirling, and M. Tofte, editors, *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 133–166. MIT Press, 2000. Also Technical Report TR-27/96, Dipartimento di Informatica, Università di Pisa, 1996.
13. M. Hennessy and H. Lin. Symbolic bisimulations. *Theoret. Comput. Sci.*, 138:353–389, 1995.
14. K.G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. In M.S. Paterson, editor, *Proceedings of ICALP'90, 17th International Colloquium on Automata, Languages and Programming*, volume 443 of *Lect. Notes in Comput. Sci.*, pages 526–539. Springer Verlag, 1990.
15. J.J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In C. Palamidessi, editor, *Proceedings of CONCUR 2000, 11th International Conference on Concurrency Theory*, volume 1877 of *Lect. Notes in Comput. Sci.*, pages 243–258. Springer Verlag, 2000.
16. R. Milner, J. Parrow, and J. Walker. A calculus of mobile processes, I and II. *Inform. and Comput.*, 100(1):1–40,41–77, 1992.
17. A. Rensink. Bisimilarity of open terms. *Inform. and Comput.*, 156(1-2):345–385, 2000.
18. D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, LFCS, University of Edinburgh, 1993. CST-99-93 (also published as ECS-LFCS-93-266).
19. D. Sangiorgi. A theory of bisimulation for the $\pi$-calculus. *Acta Inform.*, 33:69–97, 1996.
20. V. Sassone and P. Sobocinski. Deriving bisimulation congruences: A 2-categorical approach. In *Proc. of EXPRESS'02*, volume 68 of *Electronic Notes in Computer Science*. Elsevier, 2002.
21. P. Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*, 1-2(274):183–230, 2002.