Syntactic Action Refinement in the Modal Mu-Calculus and its Applications to the Verification of Reactive Systems

Mila Majster-Cederbaum, Frank Salger
Universität Mannheim
Fakultät für Mathematik und Informatik,
D7, 27, 68131 Mannheim, Germany
{mcb, fsalger}@pi2.informatik.uni-mannheim.de

Abstract: *Process calculi* like TCSP [BHR84] provide a framework in which reactive systems can be developed in a formal way. Uninterpreted *atomic actions*, viewed as the conceptual entities at a chosen level of abstraction are the basic building blocks in such calculi. The concept of *syntactic action refinement* supports the hierarchical top down developement of systems: Given a process expression P one refines an atomic action a of P by a more complex process expression Q, gaining a more detailed process expression $P[a \rightsquigarrow Q]$. *Logical calculi* (see e.g. [Eme90]) can be used as specification languages in which properties of reactive systems are denoted by formulas. The impact of action refinement on various notions of bisimulation is well understood (see e.g. [AH91, GGR94]). However little attention has been given so far to the meaning of action refinement for specification logics. In particular knowing $P \models \varphi$ does not provide information which formulas ψ are satisfied by $P[a \rightsquigarrow Q]$ or vice versa. Such knowledge however, could be used to facilitate the verification task in a stepwise refinement of systems. We extend the *Modal Mu-Calculus* [Koz83] $\mu\mathcal{L}$ by an action refinement operator and provide a link between (syntactic) action refinement for a *TCSP*-like process algebra and action refinement in $\mu\mathcal{L}$ by investigating restrictions on processes P, Q and formulas φ which guarantee

$$P \models \varphi \Leftrightarrow P[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q] \qquad (1)$$

where a is taken from a fixed set of atomic actions. Assertion (1) can be interpreted in various ways. Firstly we may interpret (1) as a simplification of the verification task, as we may check $P \models \varphi$ instead of $P[a \rightsquigarrow Q] \models \varphi[a \rightsquigarrow Q]$. Secondly we may read (1) from left to right, which embodies two views of verification: Firstly we can focus on the refinement of the specification φ. Given $P \models \varphi$, a refinement $\cdot[\alpha \rightsquigarrow Q]$ on φ automatically supplies a refined process term which satisfies the specification $\varphi[\alpha \rightsquigarrow Q]$. This can be seen as a concept of 'a-priori'-verification. Secondly we can focus on the refinement of process terms,

i.e. we consider the refinement $P[\alpha \rightsquigarrow Q]$ of P, where $P \models \varphi$ and obtain automatically $P[\alpha \rightsquigarrow Q] \models \varphi[\alpha \rightsquigarrow Q]$. One might argue that we could determine $P[\alpha \rightsquigarrow Q]$, search a specification ψ which reflects the refinement $\cdot[\alpha \rightsquigarrow Q]$ in the logic and then apply a *model checker* to establish $P[\alpha \rightsquigarrow Q] \models \psi$. However no hint is available which formula ψ reflects the refinement of P in an appropriate way. Hence we would have to test a sequence of formulas (of which we think that they reflect the refinement) $\psi_1, \psi_2, \ldots, \psi_n$ with the model checker. An application of the model checker however, might need exponential time[1]. The formula $\mathcal{R}ed(\varphi[\alpha \rightsquigarrow Q])$ which is obtained from $\varphi[\alpha \rightsquigarrow Q]$ by removing all occurrences of the refinement operator via reduction may be of size exponential in the size of Q. However only one such exponential reduction invoked by the application of assertion (1) is necessary to obtain a formula which reflects the refinement appropriatly. Hence (1) can be used as a more efficient indicator of the logical consequences induced by refinements on process terms than model checking. The validity of (1) has another interesting implication: The assertion can be used by a system designer, who is not interested in full equality of refined process expressions modulo bisimulation equivalence, but in the fact, that two refined processes both satisfy a refined property (or a set of refined properties) of special interest.

References

[AH91] L. Aceto and M. Hennessy. Adding action refinement to a finite process algebra. *Lecture Notes in Computer Science*, 510:506–519, 1991.

[BHR84] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.

[BS99] Olaf Burkart and Bernhard Steffen. Model checking the full modal mu-calculus for infinite sequential processes. *Theoretical Computer Science*, 221(1–2):251–270, June 1999.

[CS92] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. In Kim G. Larsen and Arne Skou, editors, *Proceedings of Computer Aided Verification (CAV '91)*, volume 575 of *LNCS*, pages 48–58, Berlin, Germany, July 1992. Springer.

[Eme90] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 996–1072, Amsterdam, 1990. Elsevier Science Publishers.

[Esp97] Javier Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Informatica*, 34(2):85–107, 1997.

[GGR94] U. Goltz, R. Gorrieri, and A. Rensink. On syntactic and semantic action refinement. *Lecture Notes in Computer Science*, 789:385–404, 1994.

[Koz83] D. Kozen. Results on the propositional mu -calculus. *Theoretical Computer Science*, 27(3):333–354, December 1983.

[1]Though model-checking formulas of the Modal Mu-Calculus is decidable for infinite sequential processes [BS99] it already becomes undecidable for the (parallel) process language *VBPP* [Esp97]. Non-exponential model-checker for finite state systems are known only for subsets of the Modal Mu-Calculus e.g. the alternation free fragment [CS92].