

Certification for μ -Calculus with Winning Strategies

Martin Hofmann, Christian Neukirchen, Harald Rueß

Lehr- und Forschungseinheit für Theoretische Informatik
Institut für Informatik
Ludwig-Maximilians-Universität München

SPIN 2016 · Eindhoven · Apr 8, 2016



Roadmap

- Certified decision procedures
- Certificates for model checking and μ -calculus
- Computing certificates by fixpoint instrumentation
- Certificate checking
- Evaluation

What is certified model checking?

(Namjoshi 2001)

[I]f it is determined that a property holds, model checkers produce only the answer “yes”! This does not inspire the same confidence as a counterexample; one is forced to assume that the model checker implementation is correct.

— Kedar S. Namjoshi (2001)

- Not only check whether a temporal property holds...
- ...but also give an argument *why*.

Certified decision procedures

- Decide a property, and also compute a *certificate* of it.
 - Checking the certificate only succeeds if the result is correct
- Requirements for useful certificates:
 - Checkable by a standalone, “simpler” routine
 - Checkable efficiently (in a lower complexity class)
 - Low overhead to generate
 - Compact representation
- Examples from other areas:
 - Certified UNSAT
 - Primality tests
 - Polyhedral Array-Bound Analysis

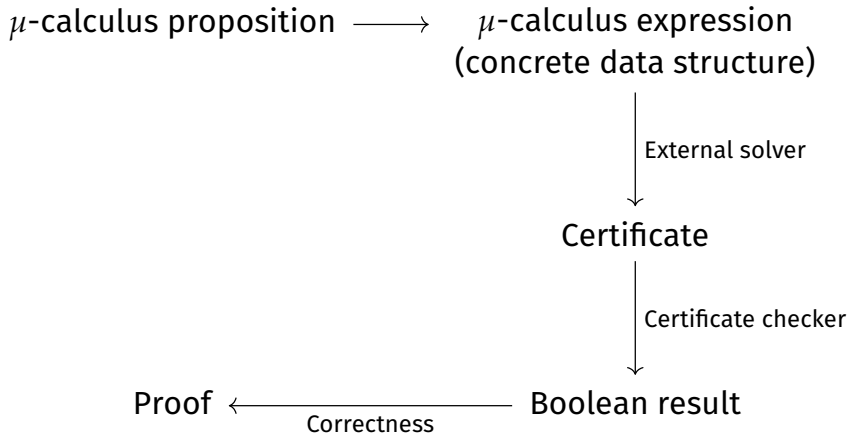
Benefits of this approach

- More confidence that the result is right
- Separation between computation and correctness
 - Can optimize decision procedure independently
- Viable for formal verification
 - Only need to verify the certificate checker
 - Trade *completeness* for implementation efficiency
 - Possible to interpret certificate as proof object
 - Possible to use in a theorem prover

Formally verified certified decision procedures

- Elegant approach: proof by reflection
- Given a temporal formula and a model in your proof logic
- Construct a concrete representation
- Use external decision procedure to compute certificate
- Interpret certificate as object of the proof logic
- Apply the correctness theorem of your certificate checker to gain a proof object of the original formula

Reflecting decision procedures with certificates



Certificates for model checking

- Familiar case: counterexamples (“negative certificate”)
 - E.g. for LTL, a *lasso* where ϕ doesn't hold to refute a $G\phi$ formula
- For simple CTL formulas, positive certificates could be:
 - for a $EF\phi$ formula, a finite path where at the end ϕ holds
 - for a $EG\phi$ formula, a lasso where ϕ holds in the loop
 - for a $AG\phi$ formula, an argument that ϕ holds for all reachable states, and all moves stay in that area
- Not obvious how certificates for general CTL or CTL* formulas look like (Shankar and Sorea 2003; Sorea 2005)
 - How to certify $AG EF \phi$?

Certificates for μ -calculus

- μ -calculus is a powerful temporal logic with arbitrary nested greatest and least fixpoint operators
 - strictly more powerful than LTL, CTL and CTL* (which are embeddable using low quantifier alternation)
 - strictly more powerful with each quantifier alternation
- Well known relationship between μ -calculus and parity games (Emerson and Jutla 1991)
- Can leverage winning strategies for parity games as certificates for μ -calculus
- Winning strategies for the dual formula ϕ^* are *generalized counterexamples*

μ -calculus and parity games

- Positions of the parity game: states \times subformulas
- Moves: according to the edges of the model resp. subformula relation
- Priorities: outer fixpoints have higher priority, μ odd, ν even, we win (= formula is true) when highest recurrent priority is even.
- Strategy: how to move at a given position
 - There are always memoryless winning strategies
 - Actual choices only on \vee and $\langle a \rangle$
 - Representable in $O(|S|^2|\phi|)$

Computing winning strategies for μ -calculus

- For finite models, μ -calculus validity is computed by fixpoint iteration in a straight forward manner.
 - As usual, computation complexity raises with quantifier alternation.
- We show that such a fixpoint computation can be *instrumented* to compute a winning strategy as well.

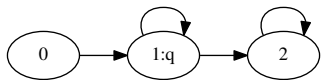
Computing winning strategies by fixpoint iteration

- Instead of computing with sets, we use *partial winning strategies*, i.e. winning strategies defined on a subset of the states.
- Inductively defined by the structure of the formula, computed in a *compositional* manner.
- For fixpoints, we iteratively grow a partial winning strategy to its maximum domain.
 - Same time complexity as computing set semantics
 - Space complexity increases from $O(|S||\phi|)$ to $O(|S|^2|\phi|^2)$ to keep track of the strategy.

Checking certificates

- Naive approach: play according to the supposed winning strategy, and branch for all possible adversarial moves. (Easily runs into exponentially many cycles.)
- Better approach: checking each SCC of the play is enough. Can reuse linear time algorithms for checking Streett automata (Duret-Lutz, Poitrenaud, and Couvreur 2009; Duret-Lutz 2007).

A small example

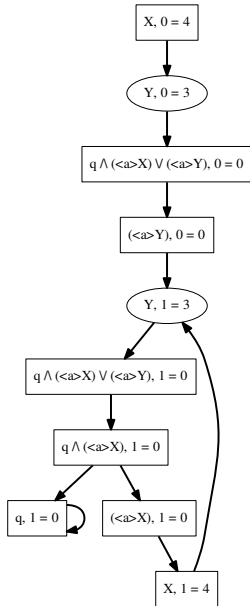


“there is a path along which q holds infinitely often”

$$\nu X. \mu Y. (q \wedge \langle a \rangle X) \vee \langle a \rangle Y$$

$$X \stackrel{\nu}{=} Y$$

$$Y \stackrel{\mu}{=} (q \wedge \langle a \rangle X) \vee \langle a \rangle Y$$

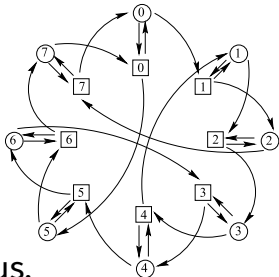


Evaluation

- Experimental implementation `micromu` in OCaml.
- Hard to find good benchmarks for μ -calculus, created three rather synthetic problems:
 - A parity game translated into μ -calculus
 - A simple reachability property to measure overhead
 - A worst-case example for checking complexity

Flower benchmark

(Buhrke, Lescow, and Vöge 1999)

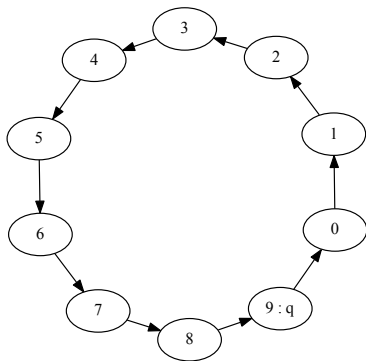


A parity game translated into μ -calculus.

With increasing problem size, solving gets exponentially harder, but checking remains polynomial.

Problem	States	sem [s]	SEM [s]	Check [s]	Check SCC [s]
Flower 8	16	0.179	0.203	0.009	0.040
Flower 10	20	3.166	1.960	0.071	0.419
Flower 12	24	32.269	11.688	0.287	2.061
Flower 14	28	320.931	61.733	1.298	10.829
Flower 16	32	3196.043	326.666	6.131	58.871

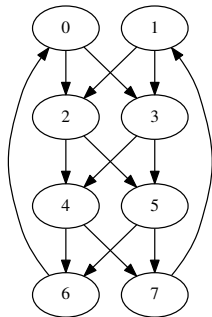
Circle benchmark



A simple reachability property
to measure overhead

Problem	States	sem [s]	SEM [s]	Check [s]	Check SCC [s]
Circle 100	100	0.003	0.001	0.001	0.001
Circle 1000	1000	0.109	0.018	0.005	0.006
Circle 10000	10000	15.763	3.398	0.054	0.057
Circle 100000	100000	2027.584	811.041	0.581	0.582

Braid benchmark



A worst-case example for checking complexity.

Naive checking blows up, but with SCC its still fine.

Problem	States	sem [s]	SEM [s]	Check [s]	Check SCC [s]
Braid 6	12	0.001	0.005	1.282	0.009
Braid 8	16	0.002	0.003	31.062	0.013
Braid 10	20	0.002	0.006	711.002	0.020
Braid 100	200	0.663	0.993	—	3.674

Perspectives

- Current implementation is pretty naive, does not use state of the art optimizations (BDDs, avoiding materialization...)
- Implementing certificate generation on existing model checkers
- Performance problems due to functional programming style
- Want to formalize certificate checking in Coq
 - ...yielding a formally verified certifying implementation of μ -calculus, usable inside a theorem prover

Questions?

Thank you.

- [1] Nils Buhrke, Helmut Lescow, and Jens Vöge. “Strategy construction in infinite games with Streett and Rabin chain winning conditions”. In: *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27-29, 1996, Proceedings*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 1055. Lecture Notes in Computer Science. Springer, 1999, pp. 207–225. ISBN: 3540610421.
- [2] Alexandre Duret-Lutz. “Contributions à l’approche automate pour la vérification de propriétés de systèmes concurrents”. PhD Thesis. Université Pierre et Marie Curie (Paris 6), July 2007.

- [3] Alexandre Duret-Lutz, Denis Poitrenaud, and Jean-Michel Couvreur. “On-the-fly Emptiness Check of Transition-based Streett Automata”. In: *ATVA'09*. Ed. by Zhiming Liu and Anders P. Ravn. Vol. 5799. Lecture Notes in Computer Science. Springer, 2009, pp. 213–227.
- [4] EA Emerson and CS Jutla. “Tree automata, mu-calculus and determinacy”. In: *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS'91)*. IEEE, 1991, pp. 368–377.
- [5] K. Namjoshi. “Certifying model checkers”. In: *Computer Aided Verification*. Springer, 2001, pp. 2–13.
- [6] N. Shankar and M. Sorea. *Counterexample-Driven Model Checking (revisited version)*. Tech. rep. SRI-CSL-03-04. SRI International, 2003.

- [7] M. Sorea. “Dubious Witnesses and Spurious Counterexamples”. UK Model Checking Days, York. 2005.