Università degli studi di Udine Laurea Magistrale: Informatica Lectures for April/May 2014 La verifica del software: temporal logic Lecture 08 Model Checking and some Advanced Topics

Guest lecturer: Mark Reynolds,
The University of Western Australia

May 15, 2014

Lecture 08

- CTL* model checking continued
- using it for LTL
- using it for CTL
- a list of some advanced topics

Proof of Correctness of Model Checking Algorithm:

Termination: clearly a finite sequence of finite operations.

Soundness: Show that every class has a fullpath at each stage. (Need to check eventualities, not just follow δ). Shows that every formula set is the set of truths of some fullpath.

Completeness. Show every fullpath has a class at each stage. By induction on k, use the actual fullpath to find the formula set that whose class it belongs to. Shows that every fullpath's true formulas are listed.

Complexity:

Computational complexity is determined as an asymptotic function of input size. For model checking the input consists of two parts: a structure and a formula.

To input the structure involves information about states and transitions. The size $|S|$ of the structure $S = (T, R, g)$ is conveniently taken to be $|S| = |T| + |R|$.

Complexity:

The size $|\phi|$ of the formula ϕ is just taken to be its length, or the number of logical symbols in the unabbreviated version.

There will be at most $|\phi|$ subformulas so at most $|\phi|$ stages in the construction.

We start with S classes and edges and each stage will at most double the number of classes and edges.

Thus there are at most $2^{|\phi|} \cdot S$ classes plus edges at any stage including the finish.

Complexity:

Each stage is, at worst, linear in the number of edges plus classes.

Thus overall complexity is $O(|\phi| \cdot 2^{|\phi|} \cdot |S|) = 2^{O(|\phi|)} \cdot O(|S|)$.

Overall it is of polynomial time complexity in the size of the structure but exponential in the size of the formula.

Model checking: theoretical complexity

A good survey of temporal logic model checking complexities can be found in [Sch02].

CTL [CES86, AC88]: $O(|S| \cdot |\phi|)$.

LTL [LP85, VW86]: $2^{O(|\phi|)} \cdot O(|S|)$.

CTL* [EL85, KVW00]: $2^{O(|\phi|)} \cdot O(|S|)$.

LTL model checking:

Not hugely popular because of the complexity issues but there exist some (industry or academic prototype) tools (that can be found online).

The most common implementations are based on symbolic model checking techniques or automata approaches.

Represent the model as a Büchi automaton. Translate the negation of the property into a Büchi automaton. Decide if the conjunction automaton is empty or not. If empty then property is satisfied.

LTL model checking via Büchi Automaton:

Translation from LTL formula to automaton is needed but is fairly straightforward.

The automaton is designed to recognise a sequence of sets of atomic propositions, corresponding to a fullpath through a structure. It should accept exactly the fullpaths that make the LTL formula true.

Easy to make the states of the (non-deterministic) automaton out of subsets of the closure set of the formula being subformulas that we want to hold from that state onwards. Need to have an acceptance criteria which checks that eventualities are fulfilled.

CTL model checking: modify the CTL* model checker

No need to ever split classes: one set of formulas per state. So no need to adjust successor relation.

But have special procedures for $AX\alpha$, $A\neg X\alpha$, $EX\alpha$, $E\neg X\alpha$, $A(\alpha U\beta)$, $A\neg(\alpha U\beta)$, $E(\alpha U\beta)$, and $E\neg(\alpha U\beta)$.

So deal with each path-temporal connective pair at once(as you move from simpler subformulas to more complicated ones).

CTL model checking:

Eg $AX\alpha$.
Put $AX\alpha$ in the label iff every successor state has α in it.

Otherwise put $\neg AX\alpha$ in.

U cases still involve more complicated searches but linear in size of structure and linear in size of formula.

CTL model checking:

Eg $A(\alpha U \beta)$.
Put $A(\alpha U \beta)$ in the label iff (a depth-first search shows) every path from here witnesses $\alpha U \beta$.

As in the CTL* model checker you do not need to allow infinite paths that do not fulfil the eventualities already in their labels.

Otherwise put $E\neg(\alpha U \beta)$ in.

Some advanced topics that follow on:

- symbolic model checking (propositional formulas describe possible states and transitions)
- bounded model checking (find a counterexample of a certain length)
- timed automata (timing constraints on transitions in system)
- metric temporal logic (timing constraints in specification)
- imperative temporal logic (program in temporal logic)

That's all:

And that's all for these lectures from me.

Good luck with your exam projects.

A. Arnold and P. Crubillé.
A linear algorithm to solve fixed-point equations on transition systems.
Information Processing Letters, 29:57–66, 1988.

E. Clarke, E. Emerson, and A. Sistla.
Automatic verification of finite-state concurrent systems using temporal logic specifications.
ACM Toplas, 8:244–263, 1986.

E. Emerson and C. Lei.
Modalities for model checking: branching time strikes back.
In *Proc. 12th ACM Symp. Princ. Prog. Lang.*, pages 84–96, 1985.

O. Kupferman, M. Vardi, and P. Wolper.
An automata-theoretic approach to branching-time model an automata-theoretic approach to branching-time model checking.

J. ACM, 47:312–360, 2000.

O. Lichtenstein and A. Pnueli.
Checking that finite state concurrent programs satisfy their linear specification.
In *Proc. 12th ACM Symp. on Princ. Prog. Lang.*, 1985.

Ph. Schnoebelen.
The complexity of temporal logic model checking.
In Philippe Balbiani, Nobu-Yuki Suzuki, Frank Wolter, and Michael Zakharyaschev, editors, *Advances in Modal Logic*, pages 393–436. King's College Publications, 2002.

M. Vardi and P. Wolper.
Automata theoretic techniques for modal logics of programs.
J. Comp. Sys. Sci., 32:183–221, 1986.