Università degli studi di Udine
Laurea Magistrale: Informatica
Lectures for April/May 2014
La verifica del software: temporal logic
Lecture 05 CTL Satisfiability via tableau

Guest lecturer: Mark Reynolds,
The University of Western Australia

May 8, 2014

Lecture 05

- finish discussing LTL sat tableau
- CTL satisfiability via tableau

LTL-SAT: What about complexity?

Deciding LTL satisfiability is in PSPACE [SC85].
In fact our tableau approach can be used to show that.
Easiest to use the tableau search to directly show that the problem is in NPSPACE and then Savitch tells us also in PSPACE.
We are allowed to guess the right choices and need to show that "yes" answers can be guessed and checked using memory space bounded by a polynomial in the size of the input. To do this, just guess the right branch and remember at each step: the label here, the previous label (to check you do the STEP rule properly), the label back at an ancestor that you want to LOOP to, and the eventualities that you still have to satisfy from that. (Size of memory usage just linear in size of input even though branch length may be exponential).

Other approaches

You might like to read up on other approaches to deciding satisfiability in LTL.
Wolper [Wol85]
Schwendiman [Sch98]
Sistla and Clarke [SC85]
Schmitt and Goubault-Larrecq [SGL97]
Automata-based approaches [VW94]
Resolution [LH10]
Others, e.g. Small model theorems with axiom systems.
Implementation competitions, and benchmarking:
[GKS10, SD11, RV07]

CTL Satisfiability via Tableau

CTL Syntax:

CTL is a restriction of CTL*.

It includes, as wffs, all atoms p, and if α and β are wffs then so are $\neg\alpha$, $\alpha \wedge \beta$, $AX\alpha$, $EX\alpha$, $A\neg X\alpha$, $E\neg X\alpha$, $A(\alpha U\beta)$, $E(\alpha U\beta)$, $A\neg(\alpha U\beta)$, and $E\neg(\alpha U\beta)$.

It uses the same semantics as CTL*.

CTL Abbreviations:

Usual classical abbreviations: $\top$, $\bot$, $\vee$, $\rightarrow$, $\leftrightarrow$.

Note that, in what follows we will regard $\neg AX\alpha$ as being the same as $E\neg X\alpha$.
Similarly, $\neg EX\alpha \equiv A\neg X\alpha$,
$\neg A\neg X\alpha \equiv EX\alpha$,
$\neg E\neg X\alpha \equiv AX\alpha$,
$\neg A(\alpha U\beta) \equiv E\neg(\alpha U\beta)$,
$\neg E(\alpha U\beta) \equiv A\neg(\alpha U\beta)$,
$\neg A\neg(\alpha U\beta) \equiv E(\alpha U\beta)$,
$\neg E\neg(\alpha U\beta) \equiv A(\alpha U\beta)$.

The usual temporal abbreviations EF, EG, AF, AG are allowed as $EF\alpha \equiv E(\top U\alpha)$ etc.

Reminder of CTL* Semantics:

Write $M, \sigma \models \alpha$ iff the formula α is true of the fullpath σ in the structure $M = (S, R, g)$ defined recursively by:

$M, \sigma \models p$	iff	$p \in g(\sigma_0)$, for $p \in \mathcal{L}$
$M, \sigma \models \neg\alpha$	iff	$M, \sigma \not\models \alpha$
$M, \sigma \models \alpha \wedge \beta$	iff	$M, \sigma \models \alpha$ and $M, \sigma \models \beta$
$M, \sigma \models X\alpha$	iff	$M, \sigma_{\geq 1} \models \alpha$
$M, \sigma \models \alpha U \beta$	iff	there is some $i \geq 0$ such that $M, \sigma_{\geq i} \models \beta$ and for each j, if $0 \leq j < i$ then $M, \sigma_{\geq j} \models \alpha$
$M, \sigma \models A\alpha$	iff	for all fullpaths τ with $\tau_0 = \sigma_0$, $M, \tau \models \alpha$

CTL Semantics:

Recall:

It can be shown that the truth of CTL formulas in a structure only depend on the initial state of the fullpath. They are sometimes called state formulas.

That is, if $\pi_0 = \sigma_0$ then $M, \pi \models \alpha$ iff $M, \sigma \models \alpha$.

Examples:

$AG(p \rightarrow AFq)$
$EFq \rightarrow EFEGq$
$AG(p \rightarrow EXp) \rightarrow AG(p \rightarrow EGp)$

Satisfiability:

We want to invent an algorithm which solves the CTL-SAT problem. Input should be a formula from CTL. Output is "yes" or "no" depending on whether the formula is satisfiable or not.

We present a tableau approach that is loosely based on the first tableau for CTL from Emerson and Clarke 1982 [EC82]. However, this is my own version of their general idea and is not published or checked very thoroughly (so apologies for any of my mistakes that remain).

It uses a different style of tableau to the tree that we used for LTL: sometimes called a graph tableau, or bottom-up.

Main technique.:

Try to find a model of the formula by building states from labels containing subformulas .

However, we generate all possible label/states first and put all reasonable transitions between the states.

Then we repeatedly throw away parts of the structure that do not make sense.

Hopefully, if the formula is satisfiable then we are left with a model of it.

The initial structure:

Given ϕ.

Let the closure set of ϕ, $\mathbf{C}(\phi) = \{\psi, \neg\psi | \psi \subseteq \phi\}$ where $\psi \leq \phi$ means ψ is a subformula of ϕ.

$2^{\mathbf{C}(\phi)} = \{B \subseteq \mathbf{C}(\phi)\}$ is our set of possible labels.

But there are some silly labels in there ...

Initial set of all labels:

Do not use any elements $B \in 2^{\mathbf{C}(\phi)}$ that fail any of the following tests.

- For all $\alpha \in \mathbf{C}(\phi)$, exactly one of α and $\neg\alpha$ is in B.
- If $\alpha \wedge \beta \in \mathbf{C}(\phi)$ then $\alpha \wedge \beta \in B$ iff both $\alpha \in B$ and $\beta \in B$.
- If $A(\alpha U \beta) \in \mathbf{C}(\phi)$ and $\beta \in B$ then $A(\alpha U \beta) \in B$.
- If $A(\alpha U \beta) \in \mathbf{C}(\phi)$ and $\neg\beta \in B$ then $\alpha \in B$.
- If $E(\alpha U \beta) \in \mathbf{C}(\phi)$ and $\beta \in B$ then $E(\alpha U \beta) \in B$.
- If $E(\alpha U \beta) \in \mathbf{C}(\phi)$ and $\neg\beta \in B$ then $\alpha \in B$.
- If $A\neg(\alpha U \beta) \in B$ then $\neg\beta \in B$.
- If $E\neg(\alpha U \beta) \in B$ then $\neg\beta \in B$.

The set of labels left is called S_0. Note that it is best to start the tableau by just constructing only the labels in S_0.

R_0:

For each $B, D \in S_0$ say that $(B, D) \in R_0$ iff *all* the following conditions are satisfied:

- If $AX\alpha \in B$ then $\alpha \in D$.
- If $A\neg X\alpha \in B$ then $\neg\alpha \in D$.
- If $A(\alpha U\beta) \in B$ and $\neg\beta \in B$ then $A(\alpha U\beta) \in D$.
- If $E(\alpha U\beta) \in D$ and $\alpha \in B$ then $E(\alpha U\beta) \in B$.
- If $A\neg(\alpha U\beta) \in B$ and $\alpha \in B$ then $A\neg(\alpha U\beta) \in D$.
- If $E\neg(\alpha U\beta) \in D$ and $\neg\beta \in B$ then $E\neg(\alpha U\beta) \in B$

Note that these rules can each be vacuously satisfied if there are no formulas of the appropriate form.

The repeated cull:

The next stage of the tableau "building" process is sometimes called a cull. It actually consists in repeated removing of states that are clearly not part of a model.

It is iterative, or repeated, because as we remove one state it may make another one become impossible.

So we start with (S_0, R_0) and then make $(S_1, R_1), (S_2, R_2), ...$
We have $S_{i+1} \subset S_i$

The finishing conditions:

As we go through this stage we look out for two conditions which allow us to terminate the stage and finish the tableau.

If there are no states left containing ϕ then we can finish the construction and answer that ϕ is unsatisfiable (no model exists).

If ϕ still appears but we can no longer throw any states away then we can terminate and conclude that ϕ is satisfiable. There will be a model that can be found from the tableau.

How to remove state B:

At each step of the cull we find a state $B \in S_i$ that is impossible.

Then we put $S_{i+1} = S_i \setminus \{B\}$.

Put $R_{i+1} = R_i \cap (S_{i+1} \times S_{i+1})$.

Then repeat the search for an impossible state within S_{i+1}.
The rules are as follows:

Removal rules:

$B \in S_i$ can be chosen for removal iff one of the following rules apply:

EXT B has no successor via R_i.

EX $EX\alpha \in B$ but B has no successor via R_i containing α.

ENX $E\neg X\alpha \in B$ but B has no successor via R_i containing $\neg\alpha$.

AU SEE NEXT SLIDES

EU SEE NEXT SLIDES

ANU SEE NEXT SLIDES

ENU SEE NEXT SLIDES

EU rule:

If $E(\alpha U \beta) \in B$ then we need to be able to find a sequence of states in S_i starting from B and following R_i such that α is in each one until the final one and it has β in it.

Otherwise, the EU rule says that we can cull B.

The search can be done finitely because the sequence does not need to repeat states.

ENU rule:

If $E\neg(\alpha U\beta) \in B$ then we need to be able to find a sequence of states in S_i starting from B and following R_i such that $\neg\beta$ is in each one until either 1) the final one and it has $\neg\alpha$ and $\neg\beta$ in it; or 2) the sequence loops.

Otherwise, the ENU rule says that we can cull B.

The search can be done finitely because we can stop when we revisit a state.

AU rule:

If $A(\alpha U \beta) \in B$ then we need to be able to find a sub-structure (S', R') in (S_i, R_i) that satisfies the following properties.

$S' \subseteq S_i$ and $R' \subseteq R_i$.
(S', R') is a finite tree with root B (and No cycles).
Every leaf node of S' has β in it and the rest contain α.
...AND ...

AU rule continued:

For any formula $EX\gamma$ in a non-leaf of S' there is an R' successor (in S') containing γ.
For any formula $E\neg X\gamma$ in a non-leaf of S' there is an R' successor (in S') containing $\neg\gamma$.
For any formula $E(\gamma U\delta)$ and not δ in a non-leaf of S' there is an R' successor (in S') containing $E(\gamma U\delta)$.
For any formula $E\neg(\gamma U\delta)$ and not $\neg\delta$ in a non-leaf of S' there is an R' successor (in S') containing $E\neg(\gamma U\delta)$.

Otherwise, the AU rule says that we can cull B.

The search can be done finitely because their are only a finite number of choices in choosing possible sub-structures.

ANU rule:

If $A\neg(\alpha U\beta) \in B$ then we need to be able to find a sub-structure (S', R') in (S_i, R_i) that satisfies properties similar to the AU rule but...

It does not have to be a tree (loops back allowed).
Every node of S' has $\neg\beta$ in it.
Every leaf node of S' (if any) has $\neg\alpha$ in it.

That is it:

That is the tableau construction.

Termination:

The construction part will terminate as it is finite.

The culling will terminate because you stop if you can not remove anything more (or before) and you start with a finite set of labels.

Soundness:

If the tableau outputs "yes" to a formula ϕ then we need to show that ϕ has a model.

This is done by building, or unwinding, a model from the final tableau set of states (S_N, R_N).

A straightforward truth lemma does not quite work. Instead you have to build some extra parts in first repeating some wintnesses to eventualities.

Completeness:

We need to show that if the formula is satisfiable then the tableau will say yes.

This is done by showing that the culling operation never removes labels that are the true labels of any state in any actual structure.

Complexity:

The complexity is clearly exponential in the size of formula.

This agrees with the known complexity of the CTL-Sat problem [FL79].

What about satisfiability checking for CTL*?:

Automata-based techniques from [ES84]. Complexity is 2EXPTIME-complete [VS85].

Tableau by [Rey11], and a tableau plus game-theoretic techniques [FLL10].

And that's all for the fifth lecture.

See you next week for model checking.


E. Emerson and E. C. Clarke.
Using branching time temporal logic to synthesise synchronisation skeletons.
Sci. of Computer Programming, 2, 1982.


E. Emerson and A. Sistla.
Deciding branching time logic.
In *Proc. 16th ACM Symposium on Theory of Computing*, 1984.


M. Fischer and R. Ladner.
Propositional dynamic logic of regular programs.
J. Computer and System Sciences, 18:194–211, 1979.


Oliver Friedmann, Markus Latte, and Martin Lange.
A decision procedure for CTL* based on tableaux and automata.
In *IJCAR'10*, pages 331–345, 2010.


Valentin Goranko, Angelo Kyrilov, and Dmitry Shkatov.

Tableau tool for testing satisfiability in ltl: Implementation and experimental analysis.
Electronic Notes in Theoretical Computer Science, 262(0):113 – 125, 2010.
Proceedings of the 6th Workshop on Methods for Modalities (M4M-6 2009).


Michel Ludwig and Ullrich Hustadt.
Implementing a fair monodic temporal logic prover.
AI Commun., 23(2-3):69–96, 2010.


Mark Reynolds.
A tableau-based decision procedure for CTL*.
Journal of Formal Aspects of Computing, pages 1–41, August 2011.


Kristin Y. Rozier and Moshe Y. Vardi.
Ltl satisfiability checking.

In Dragan Bosnacki and Stefan Edelkamp, editors, *SPIN*, volume 4595 of *Lecture Notes in Computer Science*, pages 149–167. Springer, 2007.

A. Sistla and E. Clarke.
Complexity of propositional linear temporal logics.
J. ACM, 32:733–749, 1985.

S. Schwendimann.
A new one-pass tableau calculus for PLTL.
In Harrie C. M. de Swart, editor, *Proceedings of International Conference, TABLEAUX 1998, Oisterwijk*, LNAI 1397, pages 277–291. Springer, 1998.

Viktor Schuppan and Luthfi Darmawan.
Evaluating LTL satisfiability solvers.
In Tevfik Bultan and Pao-Ann Hsiung, editors, *ATVA'11*, volume 6996 of *Lecture Notes in Computer Science*, pages 397–413. Springer, 2011.


P. Schmitt and J. Goubault-Larrecq.
A tableau system for linear-time temporal logic.
In *TACAS 1997*, pages 130–144, 1997.


M. Vardi and L. Stockmeyer.
Improved upper and lower bounds for modal logics of programs.
In *17th ACM Symp. on Theory of Computing, Proceedings*, pages 240–251. ACM, 1985.


M. Vardi and P. Wolper.
Reasoning about infinite computations.
Information and Computation, 115:1–37, 1994.


P. Wolper.
The tableau method for temporal logic: an overview.
Logique et Analyse, 28:110–111, June–Sept 1985.