15-414: Bug Catching: Automated Program Verification

Lecture Notes on CTL Model Checking

Matt Fredrikson

Carnegie Mellon University
Lecture 23
Tuesday, April 19 2022

1 Introduction

We have seen how to model computations using Kripke structures, and how to specify temporal properties of their behaviors using Computation Tree Logic. Today, we will see how to determine whether a CTL formula is true for a given Kripke structure, a process called *model checking*. The approach that we study is notable for its simplicity and reasonable complexity, which is linear in the size of the Kripke structure and size of the formula. However, many of the structures of interest in real settings are too large even for this linear-time algorithm, and in future lectures we will study techniques for coping with this problem.

Learning goals

After this lecture, you will:

- Understand how CTL model checking is accomplished by decomposing a formula into local reasoning about a given state, and global reasoning about the paths reachable from it.
- Learn about the role of monotone fixpoints in CTL model checking, and how to compute the least and greatest fixpoints of a monotone function iteratively.
- Understand how to compute the set of states in a Kripke structure that satisfy a given CTL formula, making use of CTL axioms and iterative fixpoint computation.
- See how this can be applied to model checking on the mutual exclusion example from the previous lecture.

2 Review

We start with a refresher on Kripke structures and Computation Tree Logic.

Definition 1 (Kripke structure)**.** A *Kripke frame* $(W, \curvearrowright)$ consists of a set W with a transition relation $\curvearrowright \subseteq W \times W$ where $s \curvearrowright t$ indicates that there is a direct transition from s to t in the Kripke frame $(W, \curvearrowright)$. The elements $s \in W$ are also called states. A *Kripke structure* $K = (W, \curvearrowright, v, I)$ is a Kripke frame $(W, \curvearrowright)$ with a mapping $v : W \to 2^V$, where 2^V is the powerset of V assigning truth-values to all the propositional atoms in all states. Moreover, a Kripke structure has a set of initial states $I \subseteq W$.

A Kripke structure $K = (W, \curvearrowright, v, I)$ is called a *computation structure* if W is a finite set of states and every element $s \in W$ has at least one direct successor $t \in W$ with $s \curvearrowright t$. A (computation) *path* is an infinite sequence $s_0, s_1, s_2, s_3, \ldots$ of states $s_i \in W$ such that $s_i \curvearrowright s_{i+1}$ for all i. We will always assume that the structures used in model checking are computation structures, unless otherwise noted.

Formulas in CTL represent properties of paths that are reachable from a given state, and are thus called *state formulas*. By convention, when comparing a CTL formula to a Kripke structure, we always consider the paths reachable from the initial state. CTL formulas use the **E** (existential) and **A** (universal) *path quantifiers*, which ask whether there exists a path with a given property, or whether all paths exhibit a given property.

- $\mathbf{E}P$ is a state formula where for a given Kripke structure K we have the following:

$$K, s \models \mathbf{E}P \leftrightarrow \text{ there } \textbf{exists} \text{ a path } \pi \text{ starting at } s \text{ where } \pi \models P$$

- $\mathbf{A}P$ is a state formula where for a given Kripke structure K we have the following:

$$K, s \models \mathbf{A}P \leftrightarrow \text{ for } \textbf{all} \text{ paths } \pi \text{ starting at } s, \pi \models P$$

Path quantifiers are always paired with a *path formula*, which specifies a property over a given single path. If P is a state formula, then the following are all path formulas.

- $\mathbf{X}\, P$: The next state in the path satisfies P.
- $\mathbf{G}\, P$: All states in the path satisfy P.
- $\mathbf{F}\, P$: There exists some state on the path that satisfies P.
- $P\, \mathbf{U}\, Q$: There exists some state on the path that satisfies Q. Until then, all states satisfy P.

Putting all of this together, the semantics of the logic is shown in Definition 2.

Definition 2. In a fixed computation structure $K = (W, \curvearrowright, v)$, the truth of CTL formulas in state s is defined inductively as follows:

1. $s \models p$ iff $v(s)(p) = \mathit{true}$ for atomic propositions p

2. $s \models \neg P$ iff $s \not\models P$, i.e. it is not the case that $s \models P$
3. $s \models P \wedge Q$ iff $s \models P$ and $s \models Q$
4. $s \models \mathbf{AX}\, P$ iff all successors t with $s \curvearrowright t$ satisfy $t \models P$
5. $s \models \mathbf{EX}\, P$ iff at least one successor t with $s \curvearrowright t$ satisfies $t \models P$
6. $s \models \mathbf{AG}\, P$ iff all paths $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ satisfy $s_i \models P$ for all $i \geq 0$
7. $s \models \mathbf{AF}\, P$ iff all paths $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ satisfy $s_i \models P$ for some $i \geq 0$
8. $s \models \mathbf{EG}\, P$ iff some path $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ satisfies $s_i \models P$ for all $i \geq 0$
9. $s \models \mathbf{EF}\, P$ iff some path $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ satisfies $s_i \models P$ for some $i \geq 0$
10. $s \models \mathbf{A}[P\, \mathbf{U}\, Q]$ iff all paths $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ have some $i \geq 0$ such that $s_i \models Q$ and $s_j \models P$ for all $0 \leq j < i$
11. $s \models \mathbf{E}[P\, \mathbf{U}\, Q]$ iff some path $s_0, s_1, s_2, \ldots$ starting in $s_0 = s$ has some $i \geq 0$ such that $s_i \models Q$ and $s_j \models P$ for all $0 \leq j < i$

3 CTL Model Checking

The idea behind model checking is to exploit finiteness of the state spaces to directly compute the semantics of the formulas. Given a *finite* computation structure $K = (W, \curvearrowright, v)$ the CTL model checking algorithm computes the set of all states of K in which CTL formula ϕ is true:

$$[\![\phi]\!] \stackrel{\text{def}}{=} \{s \in W \;:\; s \models \phi\}$$

The CTL model checking algorithm for a computation structure $K = (W, \curvearrowright, v)$ computes this set $[\![\phi]\!]$ by directly following the semantics in a recursive function along the equations in this lemma.

The main hurdle that we need to overcome is what to do for operators like **F**, **G**, and **U**, which specify that certain facts must hold on states encountered arbitrarily far into the future on paths. To address this, we will use some machinery having to do with *fixpoints* of monotone functions.

Monotone fixpoints. The model checking algorithm operates over sets of states, working towards computing $[\![\phi]\!]$. When working with these sets, we adopt the same notational convention for set union $\cup$ and intersection $\cap$ as we did for logical disjunction $\vee$ and conjunction $\wedge$, namely that $\cap$ and $\wedge$ bind more closely than $\cup$ and $\vee$.

Let $\wp(W)$ denote the set of all subsets of W, and let f be a function from $\wp(W)$ to $\wp(W)$. We say that f is *monotone* if and only if it preserves subset ordering, i.e.:

$$U \subseteq V \text{ implies that } f(U) \subseteq f(V) \tag{1}$$

Note that this property is exactly like the monotone functions over real numbers, with the subset relation in place of numeric inequality.

A *fixpoint* of f is an element $Z \in \wp(W)$ that is mapped to itself by f, i.e. $f(Z) = Z$. A given function may have many fixpoints; for example, every element is a fixpoint of the identity function. Two special cases that we will make use of for model checking are the *least* and *greatest* fixpoints. We denote the least fixpoint $\mu Z.f(Z)$ to be the **unique** fixpoint that is a subset of any other fixpoint, and the greatest fixpoint $\nu Z.f(Z)$ similarly. Note that a function need not have a least or greatest fixpoint. The particular special case of the seminal Knaster-Tarski fixpoint theorem shown in Theorem 3 says that monotone functions have both, and provides a recipe for finding them.

Theorem 3 (Knaster-Tarski)**.** *Every monotone function $f : \wp(W) \to \wp(W)$ has a least and a greatest fixpoint and both can be found by iteration:*

$$\mu Z.f(Z) = \bigcup_{n \geq 1} f^n(\emptyset) \qquad \nu Z.f(Z) = \bigcap_{n \geq 1} f^n(W)$$

In Theorem 3, f^n is the n-fold composition of f. So f^{n+1} is the function mapping Z to $f(f^n(Z))$ and f^1 is f, and for example, f^3 is the function mapping Z to $f(f(f(Z)))$.

For complicated functions on infinite sets, the above unions and intersections range over more than just all natural numbers and may not be directly useful in an algorithm. But model checking is typically done when the computation structure is finite. In that case, it is entirely obvious that the union and intersection only range over finitely many natural numbers. Every time we consider an additional iteration $f^n(\emptyset)$, we either find a new state that was not in the union yet. Or we do not find such a state but then, since nothing changed, the iterate $f^{n+1}(\emptyset)$ will not find anything new either. Since there are only finitely many different states in a finite state set W of a finite computation structure, we can only find new states finitely often so that the computation terminates. The argument for the intersection is correspondingly.

3.1 The algorithm

The following lemma exploits the fact that every state has a successor in computation structures, so some next state is always defined.

Lemma 4 (Next remainders)**.** *The following are sound axioms for the computation structures of CTL:*

(EG) $\mathbf{EG}\, P \leftrightarrow P \wedge \mathbf{EX}\, \mathbf{EG}\, P$

(EU) $\mathbf{E}P\, \mathbf{U}\, Q \leftrightarrow Q \vee P \wedge \mathbf{EX}\, \mathbf{E}P\, \mathbf{U}\, Q$

To compute the set of states that satisfy a CTL formula ϕ, we apply the expansion laws in Lemma 4 directly to the set of states that satisfy subformulas of ϕ. Whenever the expansion results in the same formula being on both the left and right side of an equality, the algorithm computes a fixpoint. The main question that remains is for

which cases we should use least or greatest fixpoints. The proof of Theorem 5 sorts this matter out.

Theorem 5 (CTL model checking)**.** *In computation structures, the set $[\![\phi]\!]$ of all states that satisfy CTL formula ϕ satisfies the following equations:*

1. $[\![p]\!] = \{s \in W \ : \ v(s)(p) = \mathit{true}\}$ *for atomic propositions* p
2. $[\![\neg P]\!] = W \setminus [\![P]\!]$
3. $[\![P \wedge Q]\!] = [\![P]\!] \cap [\![Q]\!]$
4. $[\![P \vee Q]\!] = [\![P]\!] \cup [\![Q]\!]$
5. $[\![\mathbf{EX}\, P]\!] = \tau_{\mathbf{EX}}([\![P]\!])$ *using the existential successor function* $\tau_{\mathbf{EX}}()$ *defined as follows:*

$$\tau_{\mathbf{EX}}(Z) \stackrel{\text{def}}{=} \{s \in W \ : \ t \in Z \text{ for some state } t \text{ with } s \frown t\}$$

6. $[\![\mathbf{AX}\, P]\!] = \tau_{\mathbf{AX}}([\![P]\!])$ *using the universal successor function* $\tau_{\mathbf{AX}}()$ *defined as follows:*

$$\tau_{\mathbf{AX}}(Z) \stackrel{\text{def}}{=} \{s \in W \ : \ t \in Z \text{ for all states } t \text{ with } s \frown t\}$$

7. $[\![\mathbf{EF}\, P]\!] = \mu Z.([\![P]\!] \cup \tau_{\mathbf{EX}}(Z))$ *where* $\mu Z.f(Z)$ *denotes the least fixpoint* Z *of the operation* $f(Z)$*, that is, the smallest set of states satisfying* $Z = f(Z)$*.*
8. $[\![\mathbf{E}P\, \mathbf{U}\, Q]\!] = \mu Z.\big([\![Q]\!] \cup ([\![P]\!] \cap \tau_{\mathbf{EX}}(Z))\big)$

The correctness argument for the verification algorithm uses the axioms together with the insight that the respective set of states that they characterize are the *smallest* set satisfying the respective equivalence. The largest set for $\mathbf{EF}\, P$ satisfying the equivalence would simply be the entire set of states, which is futile. Likewise, the smallest set of states for $\mathbf{EG}\, P$ satisfying the equivalence in EG would simply be the empty set of states, since every state has a successor in a computation structure.

Proof of Theorem 5. The proof is *not* by induction on the number of states or on the formula because the resulting formulas are not any easier than the original formulas. Instead, it handles each equation separately. While the proof was left as an exercise originally [CES83], some cases are already proved in [CGP99], some more in [BKL08], and a much more comprehensive proof including the nontrivial case $\mathbf{A}P\, \mathbf{U}\, Q$ that uses König's lemma is in [Sch03].

The first cases immediately follow the semantics of atomic propositions, propositional operators, and $\mathbf{EX}$. The remaining cases separately argue that the solution is a fixpoint and then that it is the largest or smallest, as indicated by Theorem 5.

1. By axiom EG and case 5, the formula $\mathbf{EG}\, P$ satisfies the fixpoint equation:

$$[\![\mathbf{EG}\, P]\!] = [\![P \wedge \mathbf{EX}\, \mathbf{EG}\, P]\!] = [\![P]\!] \cap \tau_{\mathbf{EX}}([\![\mathbf{EG}\, P]\!])$$

In order to show that $[\![\mathbf{EG}\, P]\!]$ is the greatest fixpoint, consider another fixpoint $H = [\![P]\!] \cap \tau_{\mathbf{EX}}(H)$ and show that $H \subseteq [\![\mathbf{EG}\, P]\!]$ by considering any state $s_0 \in H$ and showing that $s_0 \in [\![\mathbf{EG}\, P]\!]$. Since $H \subseteq [\![P]\!]$, it is enough to show that there is a path $s_0, s_1, s_2, \ldots$ such that $s_i \in H$ for all i by induction on i, implying $s_i \models P$.

n=0: The base case follows from $s_0 \in H$.

n+1: By induction hypothesis $s_n \in H$. Thus, $s_n \in H = [\![P]\!] \cap \tau_{\mathbf{EX}}(H)$, so there is a state s_{n+1} with $s_n \curvearrowright s_{n+1}$ and $s_{n+1} \in H$.

2. By axiom EU and case 5, the formula $\mathbf{E}P\,\mathbf{U}\,Q$ satisfies the fixpoint equation:

$$[\![\mathbf{E}P\,\mathbf{U}\,Q]\!] = [\![Q \vee P \wedge \mathbf{EX}\,\mathbf{E}P\,\mathbf{U}\,Q]\!] = [\![Q]\!] \cup [\![P]\!] \cap \tau_{\mathbf{EX}}([\![\mathbf{E}P\,\mathbf{U}\,Q]\!])$$

In order to show that $[\![\mathbf{E}P\,\mathbf{U}\,Q]\!]$ is also the least fixpoint, consider another fixpoint $H = [\![Q]\!] \cup [\![P]\!] \cap \tau_{\mathbf{EX}}(H)$ and show that $[\![\mathbf{E}P\,\mathbf{U}\,Q]\!] \subseteq H$. So consider any $s_0 \in [\![\mathbf{E}P\,\mathbf{U}\,Q]\!]$ and show that $s_0 \in H$. By $s_0 \in [\![\mathbf{E}P\,\mathbf{U}\,Q]\!]$, there is a path $s_0, s_1, s_2, \ldots$ and an n such that $s_n \models Q$ and $s_j \models P$ for all $0 \leq j < n$. We prove that $s_i \in H$ for all $0 \leq i \leq n$ by backwards induction on i.

$i = n$: The base case where $i = n$ follows from $s_n \in [\![Q]\!] \subseteq [\![Q]\!] \cup [\![P]\!] \cap \tau_{\mathbf{EX}}(H) = H$.

$n - 1$: By induction hypothesis, $s_n \in H$. In order to show that $s_{n-1} \in H = [\![Q]\!] \cup [\![P]\!] \cap \tau_{\mathbf{EX}}(H)$, we use that $s_{n-1} \models P$ and that s_{n-1} has a successor $s_n \in H$. Thus, $s_{n-1} \in [\![P]\!] \cap \tau_{\mathbf{EX}}(H) \subseteq H$.

This induction ends at s_0, as there are no more predecessors in the path $s_0, s_1, \ldots$ to consider, leaving us with $s_0 \in H$.

□

Since the successor function can be computed by checking off the corresponding states along the computation structure, the only remaining question is how the least and greatest fixpoints can be computed. Note all the functions in Theorem 5 are monotone, in the sense that if their parts are true in more states then the expressions themselves are true in more states, too.

Theorem 6 (Complexity)**.** *The CTL model checking problem is linear in the size of the state space $K = (W, \curvearrowright, v)$ and in the size of the formula ϕ in the sense that it is in $O(|K| \cdot |\phi|)$ where $|K| = |W| + |\curvearrowright|$.*

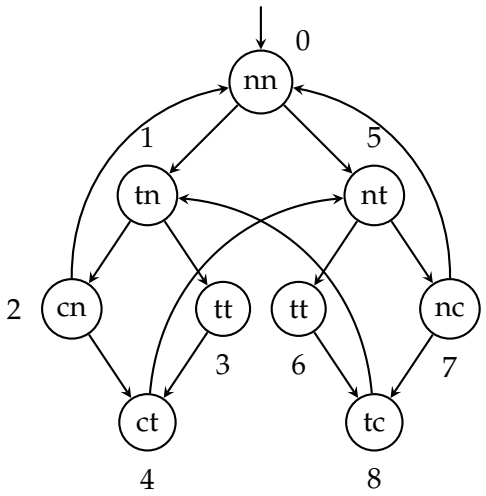
4 Example: Mutual Exclusion

Recall the mutal exclusion example introduced in the previous lecture.

The notation in the following transition diagram is nt for: the first process is in the noncritical section while the second process is trying to get into its critical section.

- n noncritical section of an abstract process
- t trying to enter critical section of an abstract process
- c critical section of an abstract process

Those atomic propositional letters are used with suffix 1 to indicate that they apply to process 1 and with suffix 2 to indicate process 2. For example the notation nt indicates a state in which $n_1 \wedge t_2$ is true (and no other propositional letters). Consider Kripke structure



1. Safety: $\neg\mathbf{EF}\,(c_1 \wedge c_2)$ is trivially true since there is no state labelled ccx.
2. Liveness: $\mathbf{AG}\,(t_1 \rightarrow \mathbf{AF}\,c_1) \wedge \mathbf{AG}\,(t_2 \rightarrow \mathbf{AF}\,c_2)$

Checking $1 \models t_1 \rightarrow \mathbf{AF}\,c_1$ alias $1 \models \neg t_1 \vee \mathbf{AF}\,c_1$ first computes subformulas.

$$
\begin{aligned}
[\![t_1]\!] &= \{1,3,6,8\} \\
[\![c_1]\!] &= \{2,4\} \\
[\![\neg t_1]\!] &= \{0,2,4,5,7\} \\
[\![\mathbf{AF}\,c_1]\!] &= \mu Z.([\![c_1]\!] \cup \tau_{\mathbf{AX}}(Z)) =: \mu Z.f(Z) \\
f^1(\emptyset) &= [\![c_1]\!] && = \{2,4\} \\
f^2(\emptyset) &= [\![c_1]\!] \cup \tau_{\mathbf{AX}}(\{2,4\}) && = \{1,2,3,4\} \\
f^3(\emptyset) &= [\![c_1]\!] \cup \tau_{\mathbf{AX}}(\{1,2,3,4\}) && = \{1,2,3,4,8\} \\
f^4(\emptyset) &= [\![c_1]\!] \cup \tau_{\mathbf{AX}}(\{1,2,3,4,8\}) && = \{1,2,3,4,6,8\} \\
f^5(\emptyset) &= [\![c_1]\!] \cup \tau_{\mathbf{AX}}(\{1,2,3,4,6,8\}) && = \{1,2,3,4,6,8\} = f^4(\emptyset) \\
[\![\mathbf{AF}\,c_1]\!] &= \{1,2,3,4,6,8\} \\
[\![\neg t_1 \vee \mathbf{AF}\,c_1]\!] &= \{0,1,2,3,4,5,6,7,8\}
\end{aligned}
$$

Since $1 \in [\![\neg t_1 \vee \mathbf{AF}\,c_1]\!]$ CTL model checking confirms $1 \models \neg t_1 \vee \mathbf{AF}\,c_1$. Since every state $[\![\neg t_1 \vee \mathbf{AF}\,c_1]\!]$ equals the set of all states, it is easy to see that model checking will also eventually find $0 \in [\![\mathbf{AG}\,(\neg t_1 \vee \mathbf{AF}\,c_1)]\!]$. Consequently it confirms that the initial state 0 satisfies $0 \models \mathbf{AG}\,(\neg t_1 \vee \mathbf{AF}\,c_1)$.

Exercises

1. Prove that the next remainder axiom for $\mathbf{G}$ is valid:

$$\text{(EG)} \quad \mathbf{EG}\, P \leftrightarrow P \wedge \mathbf{EX}\, \mathbf{EG}\, P$$

2. Identify next remainder axioms for the remaining CTL formulas not covered explicitly in this lecture: $\mathbf{EF}$, $\mathbf{AX}$, $\mathbf{AG}$, $\mathbf{AF}$, and $\mathbf{AU}$.

3. Using your axioms from the previous question, derive model checking equations for $[\![\mathbf{EF}\, P]\!]$, $[\![\mathbf{AX}\, P]\!]$, $[\![\mathbf{AG}\, P]\!]$, $[\![\mathbf{AF}\, P]\!]$, and $[\![\mathbf{A}[P\, \mathbf{U}\, Q]]\!]$.

4. Prove Theorem 5 for your solution to $[\![\mathbf{AG}\, P]\!]$ from the previous question.

References

[BKL08] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of Model Checking*. MIT Press, 2008.

[CES83] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. In *POPL*, pages 117–126, 1983.

[CGP99] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, 1999.

[Eme90] Allen Emerson. Temporal and modal logic. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. MIT Press, 1990.

[Sch03] Peter H. Schmitt. Nichtklassische Logiken. Vorlesungsskriptum Fakultät für Informatik , Universität Karlsruhe, Mai 2003.