Lecture Notes on Model Checking

15-816: Modal Logic
André Platzer

Lecture 18
March 30, 2010

1 Introduction to This Lecture

In this course, we have seen several modal logics and proof calculi to justify the truth / validity of formulas. But natural deduction, Hilbert-style calculi, tableaux, sequent calculi and the like are not the only way to establish truth.

For a particular Kripke structure at hand, a systematic exploration of the state space can be used to establish if a formula of modal logic is true or false in this Kripke structure. This process is called *model checking* [CGP99] and has been pioneered by Clarke and Emerson [CE81] and by Queille and Sifakis [QS82].

2 Model Checking $\mathbf{K}^2$

Consider the finite automaton in Figure 1 over the alphabet $\{0, 1\}$ with initial state p and accepting state F. Consider its corresponding transition structure as a Kripke structure, where the assignment of propositional letter at states is as indicated. That is, at the left-most state only propositional letter p holds, at the right-most, only s holds and so on. With this, the states of the finite automaton are captured in the Kripke structure.

The finite automaton has labels on the edges also, which cannot (really) be captured in the states. Instead, we consider a labelled transition structure where the input 0,1 is represented as labels on the accessibility relation. Now we have two accessibility relations ρ_0 and ρ_1 for the accessibility

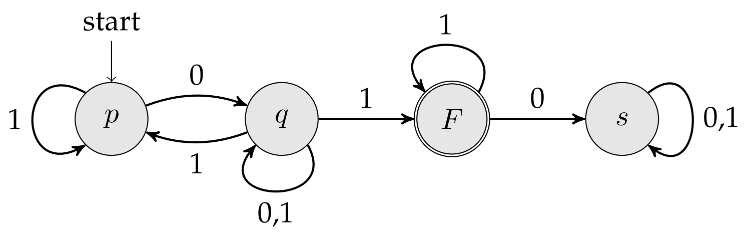

Figure 1: A finite automaton / acceptor

under input 0 and under input 1, respectively. To access these two separate accessibility relations in logical formulas, we use two separate pairs of modalities, which are also labelled with input 0 or input 1, respectively: the modality pair $\Box_0$ and $\Diamond_0$ referring to the accessibility relation ρ_0, and the modality pair $\Box_1$ and $\Diamond_1$ for the accessibility relation ρ_1.

Let K be the Kripke structure corresponding to Figure 1, then

$K \models \neg\Diamond_0 F$	does not end with 0
$K \models p \to \Diamond_0 p$	p has a 1-loop
$K \models \Diamond_0 \mathit{true}$	never stuck with input 0
$K \models \Diamond_1 \mathit{true}$	never stuck with input 1
$K \models F \to \Box_0(\neg\Diamond_0 F \land \neg\Diamond_1 F)$	no end one step after seeing 0 from F

The last formula is a bit cumbersome to write. So we introduce a third pair of modal operators $\Box_{01}$ and $\Diamond_{01}$ that we bind to refer to transition under any input (0 or 1) by assuming the following axiom (for all instantiations of formula ϕ):

$$\Diamond_{01}\phi \;\leftrightarrow\; \Diamond_0\phi \lor \Diamond_1\phi$$

With this we find that:

$K \models F \to \Box_0\neg\Diamond_{01} F$	no end one step after seeing 0 from F
$K \models F \to \Box_0\neg\Diamond_{01}\Diamond_{01} F$	no end two steps after seeing 0 from F
$K \models p \to \Diamond_{01} q$	p has a q successor
$K \models F \to \Box_1 F$	stay final on 1s

How do we establish these properties systematically? How do we model check for a given Kripke structure K and a given modal formula ϕ if $K \models \phi$? The simple-most approach is to visit each state $s \in W$ and semantically evaluate if $K, s \models \phi$. If the Kripke structure K is finite, this process terminates. It is easy to see that model checking of **K** for finite-state Kripke structures is decidable this way.

This simple process is not necessarily particularly fast, though, because the same questions may have to be answered repeatedly for some states in deeply nested modal formulas, e.g., for

$$K \models \Box_0 \Diamond_1 \Box_1 (p \vee s)$$

An alternative is to build truth-flags for all states with a computation following bottom-up dynamic programming; see Figure 2. That is, starting

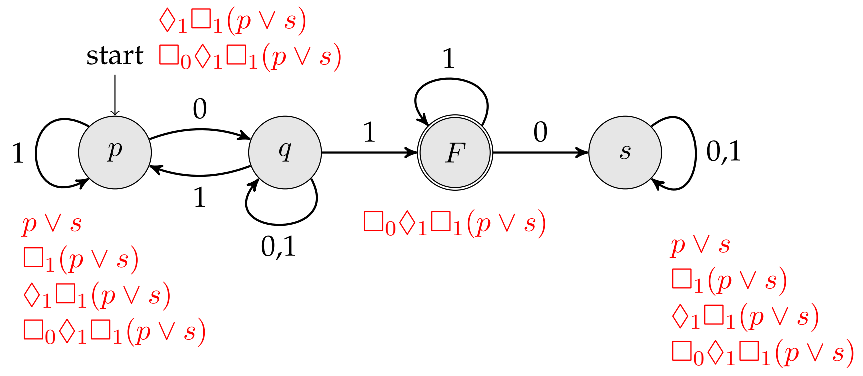
$\Diamond_1 \Box_1 (p \vee s)$
start
$\Box_0 \Diamond_1 \Box_1 (p \vee s)$
1
0
1
0
1
p
q
F
s
0,1
1
0,1
$p \vee s$
$\Box_1 (p \vee s)$
$\Diamond_1 \Box_1 (p \vee s)$
$\Box_0 \Diamond_1 \Box_1 (p \vee s)$
$\Box_0 \Diamond_1 \Box_1 (p \vee s)$
$p \vee s$
$\Box_1 (p \vee s)$
$\Diamond_1 \Box_1 (p \vee s)$
$\Box_0 \Diamond_1 \Box_1 (p \vee s)$

Figure 2: A finite automaton / acceptor

with the smallest subformulas ψ of the original formula ϕ, all states s are marked whether they satisfy ψ. Then the truth-marking for ψ is used when computing the truth of the larger subformulas of ϕ. This process solves the global model checking problem, i.e., it computes the set τ of all states of the Kripke structure where ϕ is true.

Definition 1 (Model checking problem) *Let $K = (W, \rho, v)$ be a (usually finite-state) Kripke structure, let $s \in W$ be a state of K and let ϕ be a (propositional) modal formula. Given K, s and ϕ, the problem to decide if $K, s \models \phi$ is called* local model checking problem. *Given K and ϕ, the problem of computing the set of states $s \in W$ where $K, s \models \phi$, i.e.,*

$$\tau(\phi) := \{s \in W \ : \ K, s \models \phi\}$$

is called global model checking problem.

The local model checking problem can be solved from a solution of the global model checking problem by checking if $s \in \tau(\phi)$.

With this the above global model checking process can be characterized by the following algorithm that directly follows the semantics of **K**:

$$\begin{aligned}
\tau(q) &:= \{s \in W \ : \ K, s \models \phi\} \quad \text{for a propositional letter } q \\
\tau(\neg\phi) &:= W \setminus \tau(\phi) \\
\tau(\phi \vee \psi) &:= \tau(\phi) \cup \tau(\psi) \\
\tau(\Box\phi) &:= \{s \in W \ : \ \text{all } t \text{ with } s\rho t \text{ satisfy } t \in \tau(\phi)\} \\
\tau(\Diamond\phi) &:= \{s \in W \ : \ \text{some } t \text{ with } s\rho t \text{ satisfies } t \in \tau(\phi)\}
\end{aligned}$$

Similar model checking procedures can be defined for other simple modal logics.

3 Propositional Linear Temporal Logic LTL

One canonical example for (linear time) temporal logics used prominently in model checking is the linear time temporal logic LTL [Pnu77, CGP99]. For building logical formulas LTL allows propositional letters and propositional logical operators. In addition, LTL allows modal operators:

$$\phi, \psi ::= p \mid \neg\phi \mid \phi \vee \psi \mid \phi \mathcal{U} \psi$$

Defined LTL operators give more conventional modalities

$$\begin{aligned}
\Box\phi &\equiv \mathit{true}\mathcal{U}\phi \\
\Diamond\phi &\equiv \neg(\mathit{true}\mathcal{U}\neg\phi)
\end{aligned}$$

The semantics of LTL is defined for Kripke structures with Kripke frames that are strict partial orders. Often, additional assumptions are made.

Definition 2 (Interpretation of LTL) *Given a Kripke structure $K = (W, <, v)$ that is a strict partial order, the interpretation $\models$ of LTL formulas in a world s is defined as*

1. *$K, s \models \phi\mathcal{U}\psi$ iff there is a t with $s < t$ and $K, t \models \psi$ and $K, r \models \phi$ for all r with $s < r < t$.*

4 Computation Tree Logic CTL

One canonical example for (branching time) temporal logics used prominently in model checking is the computation tree logic CTL [CGP99]. For building logical formulas CTL allows propositional letters and propositional logical operators. In addition, CTL allows modal operators:

$$\phi, \psi ::= p \mid \neg\phi \mid \phi \vee \psi \mid AX\phi \mid EX\phi \mid A(\phi \mathcal{U} \psi) \mid E(\phi \mathcal{U} \psi)$$

Definition 3 (Computation structure) *A Kripke structure $K = (W, \rho, v)$ is a* computation structure *if ρ is a partial order on W in which every state $s \in W$ has at least one direct successor t, i.e., $s\rho t$ and there is no z with $s\rho z$ and $z\rho t$. A* path *in K is an infinite sequence $s_0, s_1, \ldots, s_n, \ldots$ of states $s_n \in W$ such that s_{i+1} is a direct successor of s_i.*

A *computation tree* is a computation structure that is also a tree.

Definition 4 (Interpretation of CTL) *Given a computation structure $K = (W, \rho, v)$, the interpretation $\models$ of CTL formulas in a world s is defined as*

1. *$K, s \models A$ iff $v(s)(A) = true$.*
2. *$K, s \models \phi \wedge \psi$ iff $K, s \models \phi$ and $K, s \models \psi$.*
3. *$K, s \models \phi \vee \psi$ iff $K, s \models \phi$ or $K, s \models \psi$.*
4. *$K, s \models \neg\phi$ iff it is not the case that $K, s \models \phi$.*
5. *$K, s \models AX\phi$ iff $K, t \models \phi$ for each direct successor t of s.*
6. *$K, s \models EX\phi$ iff $K, t \models \phi$ for some direct successor t of s.*
7. *$K, s \models A(\phi \mathcal{U} \psi)$ iff for each path $s_0, s_1, \ldots, s_n, \ldots$ with $s_0 = s$ there is an $i \geq 0$ such that $K, s_i \models \psi$ and $K, s_j \models \phi$ for all j with $0 \leq j < i$.*
8. *$K, s \models E(\phi \mathcal{U} \psi)$ iff for some path $s_0, s_1, \ldots, s_n, \ldots$ with $s_0 = s$ there is an $i \geq 0$ such that $K, s_i \models \psi$ and $K, s_j \models \phi$ for all j with $0 \leq j < i$.*

When K is clear from the context, we also often abbreviate $K, s \models \phi$ by $K \models \phi$.

Beware: there are many slightly different notions of CTL. Watch out for the precise semantics. Also there are common defined CTL operators:

$$\begin{aligned}
AF\phi &\equiv A(true\mathcal{U}\phi)\\
EF\phi &\equiv E(true\mathcal{U}\phi)\\
AG\phi &\equiv \neg E(true\mathcal{U}\neg\phi)\\
EG\phi &\equiv \neg A(true\mathcal{U}\neg\phi)
\end{aligned}$$

Now we can characterize a global model checking process for CTL and finite-state computation structures by a dynamic programming computation with the least fixedpoint operator μ:

$$\begin{aligned}
\tau(q) &:= \{s \in W \ : \ K, s \models \phi\} \quad \text{for a propositional letter } q\\
\tau(\neg\phi) &:= W \setminus \tau(\phi)\\
\tau(\phi \vee \psi) &:= \tau(\phi) \cup \tau(\psi)\\
\tau(AX\phi) &:= \{s \in W \ : \ \text{all direct successors } t \text{ of } s \text{ satisfy } t \in \tau(\phi)\}\\
\tau(EX\phi) &:= \{s \in W \ : \ \text{some direct successor } t \text{ of } s \text{ satisfies } t \in \tau(\phi)\}\\
\tau(A(\phi\mathcal{U}\psi)) &:= \mu Z.\big(\tau(\psi) \cup (\tau(\phi) \cap \tau(AXZ))\big)\\
\tau(E(\phi\mathcal{U}\psi)) &:= \mu Z.\big(\tau(\psi) \cup (\tau(\phi) \cap \tau(EXZ))\big)
\end{aligned}$$

Because the respective transformation functions $t : 2^W \to 2^W$ are monotone functions on finite sets, their least fixedpoints $\mu Z.t(Z)$ can be computed by iterating from the least element (empty set):

$$\mu Z.t(Z) = \bigcup_{n\in\mathbb{N}} t^n(\emptyset)$$

We model check that $K, 1 \models T_1 \to A(true\mathcal{U}C_1)$ holds for the structure K depicted in Fig. 3. Starting from the simple subformulas we put truthmarks for all subformulas of $\neg T_1 \vee A(true\mathcal{U}C_1$ on all states.

$$\begin{aligned}
\tau(T_1) &= \{1, 4, 5, 8\}\\
\tau(C_1) &= \{3, 7\}\\
\tau(\neg T_1) &= \mathbb{Z}_8 \setminus \tau(T_1) = \{0, 2, 3, 6, 7\}\\
\tau(A(true\mathcal{U}C_1)) &= \mu Z.\tau(C_1) \cup (\tau(true) \cap \tau(AXZ)\\
&= \mu Z.\tau(C_1) \cup \tau(AXZ) =: \mu Z.t(Z)
\end{aligned}$$

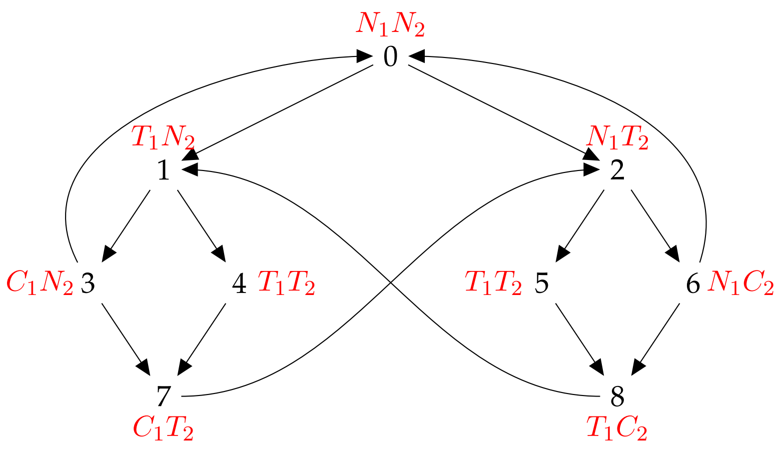

Figure 3: Mutex problem with two processes $i = 1, 2$ with states N_i=non-critical, T_i=trying, C_i=critical

We iterate to find the last fixedpoint. Let $t(Z) := \tau(C_1) \cup \tau(AXZ)$.

$$\begin{aligned}
t^0(\emptyset) &= \emptyset \\
t^1(\emptyset) &= \tau(C_1) \cup \tau(AX\emptyset) = \{3,7\} \\
t^2(\emptyset) &= \tau(C_1) \cup \tau(AX\{3,7\}) = \{3,4,7\} \\
t^3(\emptyset) &= \tau(C_1) \cup \tau(AX\{3,4,7\}) = \{1,3,4,7\} \\
t^4(\emptyset) &= \tau(C_1) \cup \tau(AX\{1,3,4,7\}) = \{1,3,4,7,8\} \\
t^5(\emptyset) &= \tau(C_1) \cup \tau(AX\{1,3,4,7,8\}) = \{1,3,4,5,7,8\} \\
t^5(\emptyset) &= \tau(C_1) \cup \tau(AX\{1,3,4,5,7,8\}) = \{1,3,4,5,7,8\}
\end{aligned}$$

Consequently we have found

$$\begin{aligned}
\tau(\neg T_1 \vee A(\mathit{true}\,\mathcal{U}C_1) &= \tau(\neg T_1) \cup \tau(A(\mathit{true}\,\mathcal{U}C_1) \\
&= \{0,2,3,6,7\} \cup \{1,3,4,5,7,8\} = \text{all states}
\end{aligned}$$

In particular, 1 is contained in the result, hence we conclude that indeed $K, 1 \models T_1 \to A(\mathit{true}\,\mathcal{U}C_1)$.

But why does this process compute the right answer?

Lemma 5 *For any computation structure, the recursive definition of $\tau(\phi)$ coincides with the set $\sigma(\phi)$ of all states at which ϕ holds true, which is defined as*

$$\sigma(\phi) := \{s \in W \;:\; K, s \models \phi\}$$

Proof: We just consider the case $E(\phi\mathcal{U}\psi))$. By induction hypothesis, we assume that $\sigma(H) = \tau(H)$ for all formulas H that are simpler than $E(\phi\mathcal{U}\psi))$. Because of the following CTL equivalence

$$E(\phi\mathcal{U}\psi) \equiv \psi \vee (\phi \wedge EXE(\phi\mathcal{U}\psi))$$

it is easy to see that the set $Z = \tau(E(\phi\mathcal{U}\psi))$ is a fixedpoint satisfying

$$Z = \tau(\psi) \cup (\tau(\phi) \cap \tau(EXZ)) \tag{1}$$

We only have to show that it is the least fixedpoint of (1). Consider another fixedpoint Z' of equation (1). By induction hypothesis, (1) is equivalent to the following because the formulas are simpler:

$$Z = \sigma(\psi) \cup (\sigma(\phi) \cap \sigma(EXZ))$$

Let $s \in Z = \tau(E(\phi\mathcal{U}\psi))$. Since $\sigma(E(\phi\mathcal{U}\psi))$ is a fixedpoint of (1) and $\tau(E(\phi\mathcal{U}\psi))$ is the smallest fixedpoint, we have $s \in \sigma(E(\phi\mathcal{U}\psi))$. That is $K, s \models E(\phi\mathcal{U}\psi)$. Hence there is a path $s_0, s_1, \ldots, s_i, \ldots$ with $s_0 = s$ and there is an $n \geq 0$ such that $s_n \in \tau(\psi)$ and $s_j \in \tau(\phi)$ for all j with $0 \leq j < i$. We show by induction on n that $s \in Z'$.

0. $n = 0$: Because Z' is a fixedpoint of (1), we have

$$Z' = \tau(\psi) \cup (\tau(\phi) \cap \tau(EXZ')) \supseteq \tau(\psi) \ni s$$

- Assume that the property holds for $n-1$ and we show that it holds for n. By induction hypothesis we have that $s_1 \in Z'$. Now because $s_0 \in \tau(\phi)$ and $s_0 \rho s_1$ we know that $s_0 \in \tau(\phi) \cap \tau(EXZ') \subseteq Z'$. Hence $s = s_0 \in Z'$.

The case $A(\phi\mathcal{U}\psi))$ is even more interesting. It is again very easy to see that the set $Z = \tau(E(\phi\mathcal{U}\psi))$ is a fixedpoint of

$$Z = \tau(\psi) \cup (\tau(\phi) \cap \tau(AXZ)) \tag{2}$$

We only have to show that it is the smallest fixedpoint. Consider another fixedpoint Z' of equation (2). Let $s \in Z = \tau(A(\phi\mathcal{U}\psi))$. As above, this implies $K, s \models A(\phi\mathcal{U}\psi)$, because $\sigma(A(\phi\mathcal{U}\psi))$ is a fixedpoint of (2) and Z is the smallest. Hence for all paths $\pi^i = s_0^i, s_1^i, \ldots, s_j^i, \ldots$ with $s_0^i = s$ there is an $n^i \geq 0$ such that $s_{n^i}^i \in \tau(\psi)$ and $s_j^i \in \tau(\phi)$ for all j with $0 \leq j < n^i$. That looks like a lot of paths. Are there finitely many or infinitely many paths?

The clue is that there are only finitely many path prefixes to consider! For each path π^i only the finite prefix until n^i is relevant. If there were infinitely many such finite prefixes then there must be infinitely many nodes in the computation structure. By König's lemma, each infinite *finite-branching* tree has an infinite path. Since there are only finitely many path prefixes (say p prefixes), let n be the maximum length: $n = \max_{i=1}^{p} n_i$. Now we can prove $s \in Z'$ by induction on n.

0. $n = 0$: Because Z' is a fixedpoint of (2), we have

$$Z' = \tau(\psi) \cup (\tau(\phi) \cap \tau(AXZ')) \supseteq \tau(\psi) \ni s$$

- Assume that the property holds for $n-1$ and we show that it holds for n. By induction hypothesis we have that $s_1^i \in Z'$ for all p path prefixes π^i $(i = 1, \ldots, p)$. Now because $s_0 \in \tau(\phi)$ and $s_1^1, \ldots, s_1^p$ are all direct successors of s_0 we know that $s_0 \in \tau(\phi) \cap \tau(AXZ') \subseteq Z'$. Hence $s = s_0 \in Z'$.

See [CGP99, Sch03] for $AG\phi$ properties and other cases of the proof. □

Lemma 6 (CTL model checking is linear) *CTL model checking for a CTL formula ϕ of length $|\phi|$ and a computation structure $K = (W, \rho, v)$ needs $(|W| + |\rho|)|\phi|$ steps to check if $K \models \phi$.*

Proof: see [CGP99].

References

[CE81] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, pages 52–71, 1981.

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

[Pnu77] Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *Symposium on Programming*, volume 137 of *LNCS*, pages 337–351. Springer, 1982.

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