Appendix to Lecture 20: Automata and LTL Model Checking

J. Meseguer

LTL Satisfaction as Language Containment

Recall that, given a Kripke structure $\mathcal{K} = (A, \rightarrow_{\mathcal{K}}, L)$ on atomic propositions AP, and choosing an initial state $a \in A$ and an LTL formula $\varphi \in LTL(AP)$, the satisfaction relation is defined by the chain of equivalences:

$$\mathcal{K}, a \models \varphi \quad \Leftrightarrow \quad \forall \pi \in \mathit{Paths}(\mathcal{K})_a \;\; \pi \models \varphi \quad \Leftrightarrow \quad \forall \pi \in \mathit{Paths}(\mathcal{K})_a \;\; \pi; L \models \varphi.$$

If we define the set $\mathit{Traces}(\mathcal{K})_a$ of the traces of $\mathcal{K}$ from initial state a as: $\mathit{Traces}(\mathcal{K})_a = \{\pi; L \mid \pi \in \mathit{Paths}(\mathcal{K})_a$, we can rephrase the above definition of satisfaction as the simpler equivalence:

$$\mathcal{K}, a \models \varphi \quad \Leftrightarrow \quad \forall \tau \in \mathit{Traces}(\mathcal{K})_a \;\; \tau \models \varphi.$$

But we can view $\mathit{Traces}(\mathcal{K})_a$ as a *language of infinite words* on the alphabet $\mathcal{P}(AP)$. Specifically, *an infinite word on an alphabet* Λ is just a function $\tau \in [\mathbb{N} \rightarrow \Lambda]$, where we suggestively denote $[\mathbb{N} \rightarrow \Lambda]$ as Λ^ω (ω denotes the set of natural numbers viewed as an "ordinal set" with its $<$ order), to emphasize that this is the language of infinte words on Λ, just as Λ^* is the language of finite words on Λ. Therefore, we have the language containment: $\mathit{Traces}(\mathcal{K})_a \subseteq \mathcal{P}(AP)^\omega$.

Now observe that the relation $\tau \models \varphi$ between an infinite word $\tau \in \mathcal{P}(AP)^\omega$ and an LTL formula $\varphi \in LTL(AP)$ is defined *independently of any Kripke structure*, since the inductive semantic definition of $\tau \models \varphi$ is given in terms of the syntactic structure of φ and can be *expressed in terms of traces*, regardless of where such traces come from. Therefore, an LTL formula $\varphi \in LTL(AP)$ also defines a language of infinite words, namely, the set of all traces τ that satisfy φ. Call this language $\mathcal{L}(\varphi)$ the *language of* φ, i.e., $\mathcal{L}(\varphi) = \{\tau \in \mathcal{P}(AP)^\omega \mid \tau \models \varphi\}$. Using this notation, we can express the satisfaction relation $\mathcal{K}, a \models \varphi$ in an even simpler, language-theoretic way by the equivalence:

$$\mathcal{K}, a \models \varphi \quad \Leftrightarrow \quad \mathit{Traces}(\mathcal{K})_a \subseteq \mathcal{L}(\varphi).$$

The intuitive meaning is that, semantically, the property φ specifies a *set of allowable traces*, so $\mathcal{K}$ starting at a satisfies property φ iff all its traces are among those allowed by φ.

Büchi Automata and Decidability of ω-Regular Languages

Recall that *regular languages* are languages recognized by finite automata; and that Boolean operations on such languages, such as union, intersection and complement, as well as properties such as language containment or language emptiness, can be *effectively computed*, resp. *decided*, by means of automata. Thanks to the work of the Swiss mathematician Richard Büchi, finite automata on an input alphabet Λ can also recognize *ω-regular languages* as subsets of the set Λ^ω of infinite words on Λ. The definition of a finite automaton $\mathcal{A}$ on an input alphabet Λ remains the same: we specify its input alphabet, finite set of states, initial state (or set of initial

states), Λ-labeled transition relation, and a subset of final/accepting states. *The only thing that changes is the notion of acceptance.* A finite word $w \in \Lambda^*$ is accepted by an automaton $\mathcal{A}$ iff the input w can reach a state in the set F of accepting states of $\mathcal{A}$. Instead, $\mathcal{A}$ will accept an infinite word $\tau \in \Lambda^\omega$ iff $F \cap \mathit{inf}(\tau) \neq \varnothing$, where $\mathit{inf}(\tau)$ denotes the set of states of $\mathcal{A}$ that are visited infinitely often by the infinite input τ. Although automata remain the same, when this new interpretation of input acceptance is given to them, they are called *Büchi automata*, in honor of Richard Büchi.

For our current purposes, we just need to use two facts about Büchi automata and ω-regular languages: (1) (**Language Intersection**) If two ω-regular languages, L_1 and L_2 on Λ are respectively recognized by Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$, then their intersection $L_1 \cap L_2$ is also an ω-regular language recognized by a Büchi automaton $\mathcal{A}_1 \otimes \mathcal{A}_2$ called the *synchronous product* of $\mathcal{A}_1$ and $\mathcal{A}_2$ (see [1], Section 9.2 for a detailed construction of $\mathcal{A}_1 \otimes \mathcal{A}_2$). (2) (**Language Emptiness**) Given a Büchi automaton $\mathcal{A}$, there is an algorithm to effectively decide whether the language $\mathcal{L}(\mathcal{A})$ recognized by $\mathcal{A}$ is empty or not. Specifically, the procedure deciding the ω-regular language emptiness problem answers "empty" when $\mathcal{L}(\mathcal{A})$ is empty, but in case $\mathcal{L}(\mathcal{A})$ is non-empty, it effectively computes a *witness* $\tau \in \mathcal{L}(\mathcal{A})$ proving its non-emptiness.

Model Checking LTL Properties with Büchi Automata

We now have almost all the ingredients needed to obtain a model checking *decision procedure* for deciding the LTL satisfaction problem $\mathcal{K}, a \models \varphi$ in case the set of states $\mathit{Reach}_{\mathcal{K}}(a)$ reachable from a is *finite*, except for two remaining technical details.

First, we need to associate to $(\mathcal{K}, a)$ a Büchi automaton $\mathbb{B}(\mathcal{K}, a)$ such that $\mathcal{L}(\mathbb{B}(\mathcal{K}, a)) = \mathit{Traces}(\mathcal{K})_a$. This is easy: we can build $\mathbb{B}(\mathcal{K}, a)$ with input alphabet $\mathcal{P}(AP)$ so that it exactly mimics the behavior of $\mathcal{K}$ from the initial state a as follows: (1) its set of states *and* its set of accepting states are both $\{\iota\} \uplus \mathit{Reach}_{\mathcal{K}}(a)$, (2) its initial state is the new added state ι, and (3) its labeled transition relation is the union:

$$\{\iota \stackrel{L(a)}{\rightarrow} a\} \cup \{b \stackrel{L(c)}{\rightarrow} c \mid b, c \in \mathit{Reach}_{\mathcal{K}}(a) \wedge b \rightarrow_{\mathcal{K}} c\}.$$

The equality $\mathcal{L}(\mathbb{B}(\mathcal{K}, a)) = \mathit{Traces}(\mathcal{K})_a$ follows trivially from this construction, since there is a one-to-one correspondence between the infinite executions of $\mathcal{K}$ from a and the infinite computations of $\mathbb{B}(\mathcal{K}, a)$ having the exact same traces by construction.

Second, we need to observe the fact that the language $\mathcal{L}(\varphi)$ is ω-regular. This is because $\mathcal{L}(\varphi)$ is the language recognized by a Büchi automaton $\mathbb{B}_\varphi$ that can be effectively constructed from the LTL formula φ. Since the details of the construction $\varphi \mapsto \mathbb{B}_\varphi$ are rather involved, I refer to Section 9.4 of [1] (or, alternatively, to Section 6.8 of [4]), where this construction is described in full detail.

We are now ready to prove the main theorem of this Appendix:

Theorem (Decidability of LTL Model Checking). When the set of states $\mathit{Reach}_{\mathcal{K}}(a)$ reachable from state a is finite, the LTL satisfaction problem $\mathcal{K}, a \models \varphi$ is decidable. Furthermore, when $\mathcal{K}, a \not\models \varphi$, the decision procedure returns a (finite representation of) a trace $\tau \in \mathit{Traces}(\mathcal{K})_a$ such that $\tau \not\models \varphi$.

Proof: Since we have the equivalence $\mathcal{K}, a \models \varphi \;\Leftrightarrow\; \mathit{Traces}(\mathcal{K})_a \subseteq \mathcal{L}(\varphi)$, we just need to have a decision procedure for effectively checking the set containment $\mathit{Traces}(\mathcal{K})_a \subseteq \mathcal{L}(\varphi)$.

But this is equivalent to checking the emptiness problem $\mathit{Traces}(\mathcal{K})_a \cap \mathcal{L}(\varphi)^c = \varnothing$, where $\mathcal{L}(\varphi)^c$ denotes the complement of $\mathcal{L}(\varphi)$ in $\mathcal{P}(AP)^\omega$. But by the semantic definition $\tau \models \neg\varphi \Leftrightarrow_{def} \tau \not\models \varphi$, we have the language identity $\mathcal{L}(\varphi)^c = \mathcal{L}(\neg\varphi)$. So we just need a decision procedure for the emptiness problem $\mathit{Traces}(\mathcal{K})_a \cap \mathcal{L}(\neg\varphi) = \varnothing$. But this is just the emptiness problem $\mathcal{L}(\mathbb{B}(\mathcal{K}, a)) \cap \mathcal{L}(\mathbb{B}_{\neg\varphi}) = \varnothing$; that is, the Büchi automata language emptiness problem $\mathcal{L}(\mathbb{B}(\mathcal{K}, a) \otimes \mathbb{B}_{\neg\varphi}) = \varnothing$, which is decidable and returns a trace $\tau \in \mathit{Traces}(\mathcal{K})_a$ if the intersection is non-empty, as desired. □

Further Reading

The already cited Chapter 9 of [1] contains a detailed description of all the concepts presented here. In particular, Section 9.5 describes an *on the fly LTL model checking algorithm* to efficiently decide the emptiness problem $\mathcal{L}(\mathbb{B}(\mathcal{K}, a) \otimes \mathbb{B}_{\neg\varphi}) = \varnothing$ using double depth first search. This is the explicit-state model checking algorithm used by both the Spin model checker [3] and the Maude LTL model checker [2]. Another useful reference for the automata-theoretic approach to model checking is provided in Chapters 5 and 6 of [4].

References

[1] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2001.

[2] S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL model checker. *Electron. Notes Theor. Comput. Sci.*, 71:162–187, 2002.

[3] G. Holzmann. *The Spin Model Checker - Primer and Reference Manual*. Addison-Wesley, 2003.

[4] D. A. Peled. *Software Reliability Methods*. Texts in Computer Science. Springer, 2001.