Appendix to Lecture 21: Automata and LTL Model Checking

J. Meseguer

LTL Satisfaction as Language Containment

Recall that, given a Kripke structure $\mathcal{Q} = (Q, \to_{\mathcal{Q}}, \mathcal{A})$ on atomic propositions Π, and choosing an initial state $q \in Q$ and an LTL formula $\varphi \in LTL(\Pi)$, the satisfaction relation is defined by the chain of equivalence:

$$\mathcal{Q}, q \models_{LTL} \varphi \quad \Leftrightarrow \quad \forall \pi \in Paths(\mathcal{Q}^{\bullet})_q \;\; \pi; preds \models_{LTL} \varphi \quad \Leftrightarrow \quad \forall \tau \in Tr(\mathcal{Q}^{\bullet})_q \;\; \tau \models_{LTL} \varphi.$$

But we can view $Tr(\mathcal{Q}^{\bullet})_q$ as a *language of infinite words* on the alphabet $\mathcal{P}(\Pi)$. Specifically, *an infinite word on an alphabet* Λ is just a function $\tau \in [\mathbb{N} \to \Lambda]$, where we suggestively denote $[\mathbb{N} \to \Lambda]$ 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 *infinite* words on alphabet Λ, just as Λ^* is the language of *finite words* on Λ. Therefore, we have the language containment: $Tr(\mathcal{Q}^{\bullet})_q \subseteq \mathcal{P}(\Pi)^{\omega}$.

Now observe that the relation $\tau \models_{LTL} \varphi$ between an infinite word $\tau \in \mathcal{P}(\Pi)^{\omega}$ and an LTL formula $\varphi \in LTL(\Pi)$ is defined *independently of any Kripke structure*, since the inductive semantic definition of $\tau \models_{LTL} \varphi$ 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 $\varphi \in LTL(\Pi)$ 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., $\mathcal{L}(\varphi) = \{\tau \in \mathcal{P}(\Pi)^{\omega} \mid \tau \models_{LTL} \varphi\}$. Using this notation, we can express the satisfaction relation $\mathcal{A}, a \models_{LTL} \varphi$ in an even simpler, language-theoretic way by the equivalence:

$$\mathcal{Q}, q \models_{LTL} \varphi \quad \Leftrightarrow \quad Tr(\mathcal{Q}^{\bullet})_q \subseteq \mathcal{L}(\varphi).$$

The intuitive meaning is that, semantically, the property φ specifies a *set of allowable traces*, so $\mathcal{Q}$ starting at q satisfies property φ iff all traces of $\mathcal{Q}^{\bullet}$ from q 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[1] $\mathbf{B}$ on an input alphabet Λ remains the same: we specify its input alphabet Λ, finite set B of states, initial state $init \in B$,

[1]See Def. 5 in §7.2 of *STACS*, where Λ is denoted L and is called the labeled set. But here we need two more pieces of information: In *STACS*, $\mathbf{B}$ is a triple $\mathbf{B} = (B, \Lambda, \to_{\mathbf{B}})$, with B a finite set; but here $\mathbf{B}$ is a 5-tuple $\mathbf{B} = (B, init, \Lambda, \to_{\mathbf{B}}, F)$, with $init \in B$ the initial state, and $F \subseteq B$ the set of *accepting states*.

Λ-labeled transition relation $\rightarrow_{\mathbf{B}}$, and subset $F \subseteq B$ 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 $\mathbf{B}$ iff the input word w can reach a state in the set F of accepting states of $\mathbf{B}$. Instead, $\mathbf{B}$ will accept an infinite word $\tau \in \Lambda^\omega$ (with $\tau(0) = init$) describing an infinite computation of $\mathbf{B}$ iff some state in F is *visited infinitely often* by τ, i.e., iff $F \cap inf_{\mathbf{B}}(\tau) \neq \emptyset$, where $inf_{\mathbf{B}}(\tau) =_{def} \{b \in B \mid |\{n \in \mathbb{N} \mid \tau(n) = b\}| = \omega\}$. Given any set A, the notation $|A| = \omega$ just abbreviates the fact that there is a *bijective* function $f : A \to \mathbb{N}$. That is, $inf_{\mathbf{B}}(\tau)$ is the set of states of $\mathbf{B}$ that are *visited infinitely often* by the infinite input word 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 $\mathbf{B}_1$ and $\mathbf{B}_2$, then their intersection $L_1 \cap L_2$ is also an ω-regular language recognized by a Büchi automaton $\mathbf{B}_1 \otimes \mathbf{B}_2$ called the *synchronous product* of $\mathbf{B}_1$ and $\mathbf{B}_2$ (see §9.2 of [1] for a detailed construction of $\mathbf{B}_1 \otimes \mathbf{B}_2$). (2) (**Language Emptiness**) Given a Büchi automaton $\mathbf{B}$, there is an algorithm to effectively decide whether the language $\mathcal{L}(\mathbf{B})$ recognized by $\mathbf{B}$ is empty or not. Specifically, the procedure deciding the ω-regular language emptiness problem answers "empty" when $\mathcal{L}(\mathbf{B})$ is empty, but in case $\mathcal{L}(\mathbf{B})$ is non-empty, it effectively computes[2] a *witness* $\tau \in \mathcal{L}(\mathbf{B})$ 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 $\mathcal{Q}, q \models \varphi$ in case the set $Reach_{\mathcal{Q}}(q)$ of states reachable from q is *finite*, except for two remaining technical details.

First, we need to associate to $(\mathcal{Q}^\bullet, q)$ a Büchi automaton $\mathbf{B}(\mathcal{Q}^\bullet, q)$ such that $\mathcal{L}(\mathbf{B}(\mathcal{Q}^\bullet, q)) = Tr(\mathcal{Q}^\bullet)_q$. This is easy: we can build $\mathbf{B}(\mathcal{Q}^\bullet, q)$ with input alphabet $\mathcal{P}(\Pi)$ so that it exactly mimics the behavior of $\mathcal{Q}^\bullet$ from the initial state q as follows: (1) its set of states *and* its set of accepting states are both $\{\iota\} \uplus Reach_{\mathcal{Q}}(q)$, (2) its initial state is the new added state ι, and (3) its labeled transition relation is the union:

$$\{\iota \overset{preds(q)}{\rightarrow} q\} \cup \{q' \overset{preds(q'')}{\rightarrow} q'' \mid q', q'' \in Reach_{\mathcal{Q}}(q) \wedge q' \rightarrow_{\mathcal{Q}^\bullet} q''\}.$$

The equality $\mathcal{L}(\mathbf{B}(\mathcal{Q}^\bullet, q)) = Tr(\mathcal{Q}^\bullet)_q$ follows trivially from this construction, since there is a one-to-one correspondence between the infinite executions of $\mathcal{Q}^\bullet$ from q and the infinite computations of $\mathbf{B}(\mathcal{Q}^\bullet, q)$ 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 $\mathbf{B}_\varphi$ that can be effectively constructed from the LTL formula φ. Since the details of the construction $\varphi \mapsto \mathbf{B}_\varphi$ are somewhat 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:

[2] The reader might wonder how τ, being an infinite object, can be effectively specified. The reason is that the set B of states is *finite*. Therefore, τ, viewed as an infinite path on a finite graph, will necessarily have *cycles*, allowing a *finite* cycle description of τ.

Theorem (Decidability of LTL Model Checking). When the set of states $Reach_{\mathcal{Q}}(q)$ reachable from state q is finite, the LTL satisfaction problem $\mathcal{Q}, q \models_{LTL} \varphi$ is decidable. Furthermore, when $\mathcal{Q}, q \not\models_{LTL} \varphi$, the decision procedure returns a (finite representation of) a trace $\tau \in Tr(\mathcal{Q}^\bullet)_q$ such that $\tau \not\models_{LTL} \varphi$.

Proof: Since we have the equivalence $\mathcal{Q}, q \models_{LTL} \varphi \;\Leftrightarrow\; Tr(\mathcal{Q}^\bullet)_q \subseteq \mathcal{L}(\varphi)$, we just need to have a decision procedure for effectively checking the set containment $Tr(\mathcal{Q}^\bullet)_q \subseteq \mathcal{L}(\varphi)$. But this is equivalent to checking the emptiness problem $Tr(\mathcal{Q}^\bullet)_q \cap \mathcal{L}(\varphi)^c = \emptyset$, where $\mathcal{L}(\varphi)^c$ denotes the complement of $\mathcal{L}(\varphi)$ in $\mathcal{P}(\Pi)^\omega$. But by the semantic definition $\tau \models_{LTL} \neg\varphi \;\Leftrightarrow_{def}\; \tau \not\models_{LTL} \varphi$, we have the language identity $\mathcal{L}(\varphi)^c = \mathcal{L}(\neg\varphi)$. So we just need a decision procedure for the emptiness problem $Tr(\mathcal{Q}^\bullet)_q \cap \mathcal{L}(\neg\varphi) = \emptyset$. But this is just the emptiness problem $\mathcal{L}(\mathbf{B}(\mathcal{Q}^\bullet, a)) \cap \mathcal{L}(\mathbf{B}_{\neg\varphi}) = \emptyset$; that is, the Büchi automata language emptiness problem $\mathcal{L}(\mathbf{B}(\mathcal{Q}^\bullet, q) \otimes \mathbf{B}_{\neg\varphi}) = \emptyset$, which is decidable and returns a "witness trace" $\tau \in Tr(\mathcal{Q}^\bullet)_q$ proving that $\tau \not\models_{LTL} \varphi$ in such a language intersection 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 $\mathcal{L}(\mathbf{B}(\mathcal{Q}^\bullet, q) \otimes \mathbf{B}_{\neg\varphi}) = \emptyset$ 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 by 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.