

Chapter 7

Model checking for PLTL

The state labeling approach as used for *PCLTL* cannot be extended to *PLTL* because it cannot be decided locally (on the information present in a state plus one transition deep around that state) whether or not to attach formulas that contain two consecutive occurrence of path operators.

Therefore a different approach is applied, based on two ideas.

The first idea is, to represent not only the system but also formulas in finite transition diagram form. A representation of a formula should then represent exactly (!) the same models, sequences of states, as make the formula true. To achieve this, the transition system has to be extended with accepting conditions, leading to the notion of *Street automaton*. The Street automaton representation of a formula can be generated in an automated fashion. The automata are then also used to represent the system, because they enable to express fairness properties (also, for a system a Street automaton can be obtained in an automated manner).

The second idea is, to take the negation of the formula that is to be checked, and construct (again in an automated manner) the product of its automaton with the automaton for the system. If this product does not accept any sequences at all (which also can be checked in an automated fashion) this means that no sequence exists that is both an execution of the system and also satisfies the negation of the formula. In that case, all sequences that are executions of the system satisfy the formula, and hence the formula holds for the system.

7.1 Satisfiability checking

A special type of automaton is used that enables a natural representation of the eventuality properties.

Definition 7.1.1 *A Street automaton over PV is a five-tuple $A = (W, \rightarrow, I, F, V)$, where W is the set of nodes, $\rightarrow \subseteq W \times W$ is the transition relation, $I \subseteq W$ is the set of initial nodes, $F \subseteq 2^W \times 2^W$ is the set of accepting conditions, and $V : W \times PV \rightarrow \{tt, ff\}$ is the valuation function.*

Definition 7.1.2 *Let A be a Street automaton s.t. $F = \{(G_j, R_j) \mid j \in J\}$. An infinite sequence of states $p = w_0 w_1 \dots$ s.t. $w_i \rightarrow w_{i+1}$ ($i \geq 0$) is accepting if $w_0 \in I$ and for each $j \in J$ either none of the states of G_j repeats infinitely often in p or some state of R_j repeats infinitely often in p .*

The tableau construction for LTL relies on associating with an LTL formula φ a Street automaton A_φ such that the accepting paths of A_φ correspond exactly to all the models for φ . The nodes of the automaton describe worlds that can appear in models for φ . Since the satisfaction of φ depends only on the valuation of the propositions appearing in φ , nodes of A_φ are sets of formulas taken from the set of the sub formulas of φ . Nodes must be consistent in the sense that they do not contain both a formula and its negation. Nodes must be also maximal in the sense that for each sub formula ϕ of φ either ϕ or $\neg\phi$ is an element of each node. The valuation of a node is consistent with the propositions contained in this node. We need to consider also nodes which do not contain φ , but rather $\neg\varphi$, as these can also appear in models for φ : if e.g. in the initial state φ holds, it need not necessarily always hold further on in the sequence. The relation between nodes is defined so as to make the next state formulas true and the until formulas potentially satisfiable. "Potentially" means that for each formula $\phi U \psi$ belonging to a node w , the transition relation will guarantee that for all paths starting at w either ϕ always holds or $\phi U \psi$ holds. The acceptance conditions of the automaton are defined such that the accepting paths are these satisfying Until formulas, i.e., that ψ is not postponed forever. This is done by defining an acceptance pair for each ψ for which $\xi U \psi$ is a sub formula of φ . For this, let $Until(\varphi) = \{\psi \mid \exists \xi : \xi U \psi \in Sub(\varphi)\}$.

Definition 7.1.3 (Decision algorithm for LTL) *A Street automaton $A_\varphi = (W, \rightarrow, I, F, V)$ over PV_φ for a formula φ is generated as follows. The sets of formulas built during the execution of the algorithm are called pre-atoms.*

1. Build pre-atoms $A = \{\varphi\}$, $A' = \{\neg\varphi\}$;

2. while there are preatoms B that contain untagged formulas and if no formula would be put into a pre-atom that already contains its negation (*), apply, in arbitrary order, one of the following rules:

- (a) if $\phi \wedge \psi \in B$, then replace B by $B \cup \{\phi, \psi\}$, tag $\phi \wedge \psi$;
- (b) if $\neg(\phi \wedge \psi) \in B$, then replace B by as many as allowed by (*), but at least one, of $B \cup \{\neg\phi, \psi\}$, $B \cup \{\phi, \neg\psi\}$, $B \cup \{\neg\phi, \neg\psi\}$, tag $\neg(\phi \wedge \psi)$.
- (c) if $\phi U \psi \in B$, then replace B by as many as allowed by (*), but at least one, of $B \cup \{\phi, \psi\}$, $B \cup \{\phi, \neg\psi\}$, $B \cup \{\neg\phi, \psi\}$, tag $\phi U \psi$.
- (d) if $\neg(\phi U \psi) \in B$, then replace B by as many as allowed by (*), but at least one, of $B \cup \{\phi, \neg\psi\}$, $B \cup \{\neg\phi, \neg\psi\}$, tag $\neg(\phi U \psi)$.
- (e) if $\bigcirc\phi \in B$, then replace B by as many as allowed by (*) of $B \cup \phi$, $B \cup \neg\phi$.
- (f) if $\neg\bigcirc\phi \in B$, then replace B by as many as allowed by (*) of $B \cup \phi$, $B \cup \neg\phi$.

An atom is a pre-atom where the only un-tagged formulas are propositions or their negations.

- W is the set of all atoms.
- I is the set of atoms containing φ .
- $V(w, p) = tt$ iff $p \in w$, for $p \in PV_\varphi$.

The transition relation \rightarrow between atoms is defined as follows: $B \rightarrow B'$ iff the following conditions hold:

1. if $\bigcirc\phi \in B$, then $\phi \in B'$,
2. if $\neg\bigcirc\phi \in B$, then $\neg\phi \in B'$,
3. if $\phi U \psi \in B$ and $\neg\psi \in B$, then $\phi U \psi \in B'$,
4. if $\neg(\phi U \psi) \in B$ and $\phi \in B$, then $\neg(\phi U \psi) \in B'$.

The set F of accepting conditions is defined as follows:

$$F = \{(G, R) \mid \exists \psi \in \text{Until}(\varphi) : G = \{B \in W \mid \exists \xi \xi U \psi \in B, \neg\psi \in B\} \neq \emptyset, \\ R = \{B \in W \mid \psi \in B\}\}.$$

A formula φ is satisfiable iff the automaton A_φ contains an accepting path.

The time complexity of generating A_φ is $O(2^{|\varphi|} \times E^2)$, where E denotes the size of $\text{Until}(\varphi)$.

Theorem 7.1.1 *The above algorithm is a decision procedure for LTL.*

The proof of this theorem uses the following lemmas. Let M be an LTL model. For all $i \geq 0$, let u_i be the subset of $Sub(\varphi)$ and its negations consisting of formulas ϕ s.t. $M, i \models \phi$. We will say that the sequence $u_0 u_1 \dots$ corresponds to model M .

Lemma 7.1.1 *For all $i \geq 0$, u_i is an atom.*

Lemma 7.1.2 *$u_0 u_1 \dots$ is an accepting path of A_φ .*

Lemma 7.1.3 *Each accepting path of A_φ corresponds to a model for φ .*

7.2 Checking a Streett automaton for non-emptiness

A Streett automaton B contains, by Definition 1.1.2, an accepting path iff B contains an infinite path of states $p = w_0 w_1 \dots$ satisfying the *Street conditions* i.e., $w_0 \in I$ and for each $j \in J$ either none of the states of G_j repeats infinitely often in p or some state of R_j repeats infinitely often in p . An intuitive idea of the algorithm is as follows. By a strongly connected component (SCC) we mean a graph with at least one edge in which every two nodes are connected by a directed path. We will only consider SCC reachable from some initial state of B . A SCC C is J' -Street for $J' \subseteq J$ if for each $j \in J'$ either $G_j \cap C = \emptyset$ or $R_j \cap C \neq \emptyset$. It is easy to notice that if C is J -Street, then it contains an accepting path and therefore B contains so.

First, the algorithm finds all maximal strongly connected components. This can be done in time linear in $|W| + |\rightarrow|$ (see [Aho, Hopcroft, Ullman]). In fact, the algorithm could have dealt with all the connected components, but this would be very inefficient as there may be exponentially many of them.

Then, for each SCC C of B , the algorithm computes the set of indices $J' \subseteq J$ such that $j \in J'$ iff C is NOT $\{j\}$ -Street. If $J' = \emptyset$, then we are done - C is J -Street. Otherwise we have to avoid nodes G_j since $C \cap R_j = \emptyset$, for $j \in J'$. One can notice that if C contains a path satisfying the Street conditions, then $C' = C - \bigcup \{G_j \mid j \in J'\}$ also contains a path satisfying the Street conditions. To see this, assume that p is a path of C satisfying the Street conditions. Then, either p does not contain $\{G_j \mid j \in J'\}$ states or contains only finitely many of them. In the first case we are done, C' contains the path p as well. In the second case, C must contain a path p' which does not contain

states from $\{G_j \mid j \in J'\}$, but contains exactly the states that repeat infinitely often in p . Thus, p' satisfies the Street conditions and C' contains the path p' as well.

Then, the algorithm is applied recursively to the subgraph C' . This process terminates because there are only finitely many SCCs and the recursion depth is bounded by $|J|$.

The time complexity of this algorithm is $O(|J| \times |W| \times |\rightarrow|)$.

The total complexity of the verification is therefore $O(|A_P| \times 2^{|\varphi|} \times E^2)$.

7.3 Model checking for LTL

To check whether or not a model represented as a Street automaton satisfies a PLTL property represented as a Street automaton the product of two Street automata is used. Essentially, this is just the well-known Cartesian product of automata. As to the acceptance conditions: intuitively, an accepting sequence should fulfil the acceptance conditions of both automata. Therefore the acceptance conditions of both components are present, adapted to the product automaton by taking the Cartesian product with all nodes of the other component. The intersection with W reflects that acceptance conditions only meaningfully apply to states of the product component.

Definition 7.3.1 (product of two Street automata) *The product $A_1 \times A_2$ of two Street automata: $A_1 = (W_1, \rightarrow_1, I_1, F_1, V_1)$ over PV and $A_2 = (W_2, \rightarrow_2, I_2, F_2, V_2)$ over PV is the automaton $B = (W, \rightarrow, I, F, V)$ over PV defined by*

- $W = \{(w_1, w_2) \in W_1 \times W_2 \mid V_1(w_1) = V_2(w_2)\},$
- $I = (I_1 \times I_2) \cap W,$
- $F = \{((G_j^1 \times W_2) \cap W, (R_j^1 \times W_2) \cap W) \mid j \in J_1\} \cup \{((W_1 \times G_j^2) \cap W, (W_1 \times R_j^2) \cap W) \mid j \in J_2\}$ where $F_1 = \{(G_j^1, R_j^1) \mid j \in J_1\}$ and $F_2 = \{(G_j^2, R_j^2) \mid j \in J_2\},$
- $(v_1, v_2) \rightarrow (w_1, w_2)$ iff $v_1 \rightarrow_1 w_1$ and $v_2 \rightarrow_2 w_2.$
- $V((w_1, w_2)) = V_1(w_1),$ for all $(w_1, w_2) \in W.$

The time complexity of this algorithm is $O(|A_1| \times |A_2|)$.

Checking whether or not a model M represented as a Street automaton A_P satisfies a PLTL property φ the negation of which is represented as a Street

automaton $A_{\neg\varphi}$ now just amounts to checking whether there is a path that is accepted by both A_P and $A_{\neg\varphi}$, i.e., whether or not the product of these Streett automata is empty.