Linear-Time Model Checking

1. LTL
2. Basic principles
   - Buchi Automata
   - LTL — Buchi Automata
   - Automata-theoretic model-checking
3. SPIN/Promela
   - expressing models
   - partial-order reductions
4. LTL + partial-order reductions
   - closure under stuttering
   - language expressiveness

Propositional Linear-Time Temporal Logic

- If $p$ is an atomic propositional formula, it is a formula in LTL.
- If $p$ and $q$ are LTL formulas, so are $p \land q$, $p \lor q$, $\neg p$, $pUq$, $\diamond p$ (next), $\diamond p$ (eventually), $\square p$ (always).

Interpretation: over computations $\pi : \omega \to 2^{\text{Prop}}$ which assigns truth values to the elements of Prop at each time instant:

- $\pi, i \models p$ for $p \in \text{Prop}$ iff $p \in \pi(i)$
- $\pi, i \models \diamond \phi$ iff $\pi, i + 1 \models \emptyset$
- $\pi, i \models \phi U \psi$ iff for some $j \geq i$, $\pi, j \models \psi$ and for all $k$, $i \leq k < j$, $\pi, k \models \phi$ (strong until)
- $\pi, i \models \square \phi$ iff for all $j \geq i$, $\pi, j \models \phi$
- $\pi, i \models \diamond \phi$ iff exists $j \geq i$, $\pi, j \models \phi$

$\pi$ satisfies a formula $\phi (\pi \models \phi)$ iff $\pi, 0 \models \phi$. 

\[ 138 \] 139
Reading Exercises

Following are some temporal formulas $\varphi$ and what they say about a sequence $\sigma : s_0, s_1, \ldots$ s.t. $\sigma \models \varphi$:

- $p \rightarrow \Diamond q$ - If $p$ holds at $s_0$, then $q$ holds at $s_j$ for some $j \geq 0$.

- $\Box (p \rightarrow \Diamond q)$ - Every $p$ is followed by a $q$.

- $\Box \Diamond q$ - The sequence $\sigma$ contains infinitely many $q$’s.

- $\Box \Diamond q$ - All but a finitely many states in $\sigma$ satisfy $q$. Property $q$ eventually stabilizes.

LTL is good for expressing safety and liveness properties:

- $\Box (p U q)$ - always $p$ remains true at least until $q$ becomes true.
- $\neg (\Diamond (p U q))$ - never is there a point in the execution s.t. $p$ remains true at least until $q$ becomes true.
- $\neg (p U (\Box (q U r)))$ - it is not true that $p$ is true at least until the point s.t. for all paths $q$ is true at least until $r$ is true.
Some Temporal Properties

\[
\begin{align*}
\Diamond p &= \text{True } \cup p & \text{-- Eventually} \\
\square p &= \neg \Diamond \neg p & \text{-- Henceforth} \\
pWq &= \square p \lor (p \square q) & \text{-- Waiting-for, Unless, } V \\
\square p &= p \land \Diamond p \\
\Diamond p &= p \lor \Diamond p \\
p\square q &= q \lor (p \land \Diamond (p \square q))
\end{align*}
\]

Comparison of LTL with CTL

Syntactically, LTL is simpler than CTL.

Semantically, the two are incomparable:

- The CTL formula EF \( p \), stating the existence of a path leading to a \( p \)-state is inexpressible in LTL.
- The LTL formula \( \Diamond \square p \) stating that every computation eventually stabilizes at \( p \) is inexpressible in CTL. The following automaton:

satisfies \( \Diamond \square p \) but does not satisfy the CTL approximation AF AG \( p \).

Most useful properties are specifiable by both. Invariance can be specified by both \( \square p \) and AG\( p \). Liveness (response) is specifiable by both \( \square (p \rightarrow \Diamond q) \) and AG\( (p \rightarrow \text{AF} q) \).
Crucial Connection: LTL $\equiv$ Buchi Automata

**Buchi Automata**

If $A$ is an alphabet, let $A^*$ denote the set of finite words, and $A^\omega$ - the set of infinite words ($\omega$-words) over $A$.

Example: $A = \{a, b\}$, $\alpha = abaabaab...$

Can define languages $L \subseteq A^\omega$ on $\omega$-words and automata that recognize such languages.

Example: $A = \{a, b, c\}$, $L_1 \subseteq A^\omega$ is $\alpha \in L_1$ iff after any occurrence of letter $a$ there is some occurrence of letter $b$ in $\alpha$.

Possible strings:
\[
abaab...  \quad abaaabab...
\]
\[
ababbabb...  \quad accbacb...
\]

Automata for recognizing such languages are called Buchi.

---

**Buchi Automata**

Definition: A Buchi automaton over the alphabet $A$ is of the form $A = (Q, q_0, \Delta, F)$ with finite state set $Q$, initial state $q_0 \in Q$, transition relation $\Delta \subseteq Q \times A \times Q$, and a set $F \subseteq Q$ of final states.

A run of $A$ on an $\omega$-word $\alpha = \alpha(0)\alpha(1)...$ from $A^\omega$ is a sequence $\delta = \delta(0)\delta(1)...$ such that $\delta(0) = q_0$ and $(\delta(i), \alpha(i), \delta(i+1)) \in \Delta$ for $i \geq 0$; the run is called *successful* if some state of $F$ occurs infinitely often in it.

- $A$ accepts $\alpha$ if there is a successful run of $A$ on $\alpha$.
- $L(A) = \{\alpha \in A^\omega \mid A$ accepts $\alpha\}$ - $\omega$-language recognizable by $A$.
- If $L = L(A)$ for some Buchi automaton $A$, $L$ is said to be Buchi-recognizable.
A Note on Notation

Vardi, Wolper

Büchi automaton \((\Sigma, S, \rho, s_0, F)\).

Thomas

Büchi automaton \((Q, q_0, \Delta, F)\) over alphabet \(A\).

Correspondences:

- \(A \equiv \Sigma\) alphabet
- \(Q \equiv S\) set of states
- \(q_0 \equiv s_0\) initial states
- \(\Delta \equiv \rho\) transition relation
- \(F \equiv F\) accepting states

Important Theoretical Results

1. The emptiness problem for Büchi automata is decidable \((L(A) \neq \emptyset)\) (logspace complete for NLOGSPACE, i.e., solvable in linear time [Vardi, Wolper])

2. Nonuniversality problem for Büchi automata \((L(A) \neq A^\omega)\) is decidable (logspace complete for PSPACE [Sisla, Vardi, Wolper])

3. Büchi automata are closed under complementation, i.e., from a Büchi automaton recognizing \(L\) one can construct an automata recognizing \(A^\omega - L\). The number of states in this automaton is \(2^{O(\|\phi\|)}\) states (\(\phi\) - formula representing language \(L\) - see later)

4. Büchi automata are closed under intersection [Chouka74]: given two Büchi automata \(A\) (with \(S\) states) and \(B\) (with \(S_1\) states), one can construct an automaton with \(2|S| \times |S_1|\) states that accepts \(L(A) \cap L(B)\)
Relationship between LTL Formula and Buchi Automata

Theorem [Wolper, Vardi, Sisla 83]: Given an LTL formula $\phi$, one can build a Buchi automaton $A_\phi = (\Sigma, S, \rho, s_0, F)$ where $\Sigma = 2^{\text{Prop}}$ (the number of automatic propositions, variables, etc. in $\phi$) and $|S| \leq 2^{O(|\phi|)}$ (length of the formula) s.t. $L(A_\phi)$ is exactly the set of computations satisfying the formula $\phi$.

Examples:

- $\Box(p \lor q)$
- $\Box \diamond p$
- $\Box \diamond (p \lor q)$
- $\neg \Box \diamond (p \lor q)$
- $\neg (\Box(p \lor q))$

Sketch of the Algorithm

Compute the set of subformulas that must hold in each reachable state and in each of its successor states.

- Convert formula into normal form (negation for atomic propositions)
- Create initial state, marked with the formula to be matched and a dummy incoming edge
- Recursively
  - take a subformula that remains to be satisfied
  - look at the leading temporal operator: may split the current state into two, each annotated with appropriate subformula
- Make connections to accepting state

More info in Vardi and Wolper’s proof.
Linear-Time TL Modelchecking

Given a finite-state program $P = (W, s_0, R, V)$ ($W$ — finite set of states, $s_0 \in W$ — initial state, $R \subseteq W^2$ — total accessibility relation, $V : W \to 2^{\text{Prop}}$ — assigns truth values to propositions in Prop for each state in $W$), we can represent it as a Buchi automaton $A_P = (2^{\text{Prop}}, W, \{s_0\}, \rho, W)$.

Here, $s_t \in \rho(s, a)$ iff $(s, s_t) \in R$.

$s_0$ — the only starting state. All states are accepting. $a = V(S)$.

So, want to know if all sequences accepted by $A_P$ are also accepted by $A_{\phi}$ (automaton equivalent to property $\phi$):

- Compute complement of $A_{\phi}$ ($\overline{L(A_{\phi})}$)
- Intersect result with $A_P$
- Check for emptiness

---

Linear-Time ModelChecking (Cont’d)

So, build an automaton for $L(A_P) \cap \overline{L(A_{\phi})}$ with $|W| \times 2^{O(|\phi|)}$ states.

Thus:

- Program complexity of the verification problem is logspace complete for co-NLOGSPACE.
- The specification complexity of the verification problem is logspace complete for PSPACE.
- Checking whether a formula $\phi$ is satisfied by a finite-state program $P$ can be done in time $O(||P|| \times 2^{O(|\phi|)})$ or in space $O((\log ||P|| \cdot \log ||\phi||)^2)$.

i.e., checking is polynomial in the size of the program and exponential in the size of the specification.
food for slide eater