Linear-Time ModelChecking

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 $p \wedge q$, $p \vee q$, $\neg p$, $p\mathcal{U}q$, $\circ p$ (next), $\diamond p$ (eventually), $\Box p$ (always).

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

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

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

Reading Exercises

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

- $p \to \diamond q$ – If p holds at s_0, then q holds at s_j for some $j \geq 0$.

- $\Box(p \to \diamond q)$ – Every p is followed by a q.

- $\Box \diamond q$ – The sequence σ contains infinitely many q's.

- $\Box \diamond q$ – All but a finitely many states in σ satisfy q. Property q eventually stabilizes.

LTL is good for expressing safety and liveness properties:

- $\Box(p\mathcal{U}q)$ - always p remains true at least until q becomes true.
- $\neg(\diamond(p\mathcal{U}q))$ - never is there a point in the execution s.t. p remains true at least until q becomes true.
- $\neg(p\mathcal{U}(\Box(q\mathcal{U}r)))$ - 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

$$
\begin{array}{rcll}
\diamond p & = & \textit{True}\ \text{U p} & \text{– Eventually} \\
\square p & = & \neg \diamond \neg p & \text{– Henceforth} \\
p\mathcal{W}q & = & \square p \vee (p\mathcal{U}q) & \text{– Waiting-for, Unless, V} \\
\square p & = & p \wedge \circ\square p & \\
\diamond p & = & p \vee \circ \diamond p & \\
p\mathcal{U}q & = & q \vee (p \wedge \circ(p\mathcal{U}q)) &
\end{array}
$$

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 AGp. Liveness (response) is specifiable by both $\square(p \rightarrow \diamond q)$ and AG$(p \rightarrow$ 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^ω - the set of infinite words (ω-words) over A.

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

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

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

Possible strings:

$ababab...$ $\quad aaabaaab...$

$abbabbabb...$ $\quad accbaccb...$

Automata for recognizing such languages are called Buchi.

Buchi Automata

Definition: A Buchi automaton over the alphabet A is of the form $\mathcal{A} = (Q, q_0, \Delta, F)$ 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 $\mathcal{A}$ on an ω-word $\alpha = \alpha(0)\alpha(1)...$ from A^ω 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.

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

A Note on Notation

Vardi, Wolper

Buchi automaton $(\Sigma, S, \rho, s_o, F)$.

Thomas

Buchi automaton (Q, q_0, Δ, F) over alphabet A.

Correspondences:

A	$\equiv$	Σ	alphabet
Q	$\equiv$	S	set of states
q_0	$\equiv$	s_0	initial states
Δ	$\equiv$	ρ	transition relation
F	$\equiv$	F	accepting states

Important Theoretical Results

1. The emptiness problem for Buchi automata is decidable ($L(\mathcal{A}) \neq \emptyset$) (logspace complete for NLOGSPACE, i.e., solvable in linear time [Vardi, Wolper])
2. Nonuniversality problem for Buchi automate ($L(\mathcal{A}) \neq A^\omega$) is decidable (logspace complete for PSPACE [Sisla, Vardi, Wolper])
3. Buchi automata are closed under complementation, i.e., from a Buchi automaton recognizing L one can construct an automata recognizing $A^\omega - L$. The number of states in this automaton is $2^{O(|\phi|)}$ states (ϕ - formula representing language L - see later)
4. Buchi automata are closed under intersection [Chouka74]: given two Buchi 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 ϕ, one can build a Buchi automaton $\mathcal{A}_\phi = (\Sigma, S, \rho, s_0, F)$ where $\Sigma = 2^{\mathsf{Prop}}$ (the number of automatic propositions, variables, etc. in ϕ) and $|S| \leq 2^{O(|\phi|)}$ ($|\phi|$ - length of the formula) s.t. $L(\mathcal{A}_\phi)$ is exactly the set of computations satisfying the formula ϕ.

Examples:

$\Box(p\mathcal{U}q)$

$\Box \diamond p$

$\Box \diamond (p \vee q)$

$\neg\Box \diamond (p \vee q)$

$\neg(\Box(p\mathcal{U}q))$

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, $V : W \rightarrow 2^{\text{Prop}}$ – assigns truth values to propositions in Prop for each state in W), we can represent it as a Buchi automaton $\mathcal{A}_P = (2^{\text{Prop}}, W, \{s_0\}, \rho, W)$.

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

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

So, want to know if all sequences accepted by $\mathcal{A}_P$ are also accepted by $\mathcal{A}_\phi$ (automaton equivalent to property ϕ):

- Compute complement of $\mathcal{A}_\phi$ ($\overline{L(\mathcal{A}_\phi)}$)
- Intersect result with $\mathcal{A}_p$
- 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 ϕ is satisfied by a finite-state program P can be done in time $O(||P|| \times 2^{O(|\phi|)})$ or in space $O((log||P|| + ||\phi||)^2)$.

i.e., checking is polynomial in the size of the program and exponential in the size of the specification.

food for slide eater

food for slide eater