

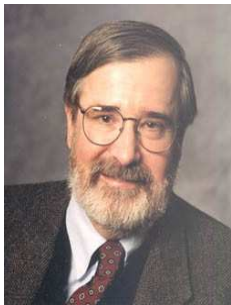
A blue crosshair graphic consisting of a vertical line and a horizontal line intersecting in the upper-left quadrant of the slide.

CS 228 : Logic in Computer Science

Krishna. S

Linear Temporal Logic

Model Checking



- ▶ Year 2007 : ACM confers the **Turing Award** to the pioneers of Model Checking: **Ed Clarke, Allen Emerson, and Joseph Sifakis**

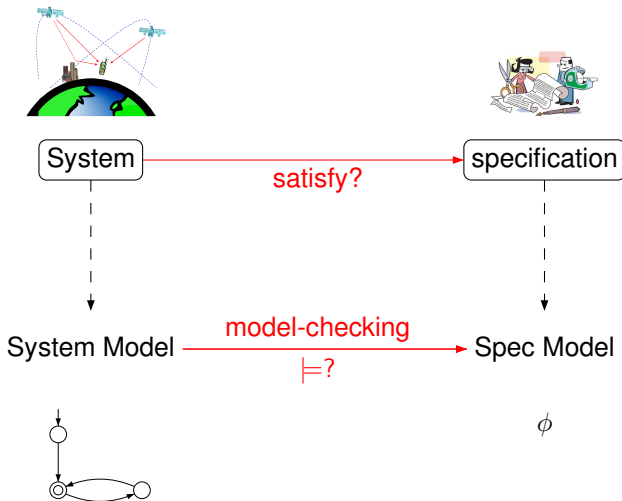


https://amturing.acm.org/award_winners/clarke_1167964

Model checking

- ▶ Model checking has evolved in last 25 years into a widely used verification and debugging technique for software and hardware.
- ▶ Model checking used (and further developed) by companies/institutes such as IBM, Intel, NASA, Cadence, Microsoft, and Siemens, and has culminated in many freely downloadable software tools that allow automated verification.

What is Model Checking?



Model Checker as a Black Box

- ▶ Inputs to Model checker : A finite state system M , and a property P to be checked.
- ▶ Question : Does M satisfy P ?
- ▶ Possible Outputs
 - ▶ Yes, M satisfies P
 - ▶ No, here is a counter example!.

What are Models?

Transition Systems

- ▶ States labeled with propositions
- ▶ Transition relation between states
- ▶ Action-labeled transitions to facilitate composition

What are Properties?

Example properties

- ▶ Can the system reach a deadlock?
- ▶ Can two processes ever be together in a critical section?
- ▶ On termination, does a program provide correct output?

Notations for Infinite Words

- ▶ Σ is a finite alphabet
- ▶ Σ^* set of finite words over Σ
- ▶ An infinite word is written as $\alpha = \alpha(0)\alpha(1)\alpha(2)\dots$, where $\alpha(i) \in \Sigma$
- ▶ Such words are called ω -words
- ▶ a^ω , $a^7.b^\omega$

Transition Systems

A **Transition System** is a tuple $(S, Act, \rightarrow, I, AP, L)$ where

- ▶ S is a set of **states**
- ▶ Act is a set of **actions**
- ▶ $s \xrightarrow{\alpha} s'$ in $S \times Act \times S$ is the **transition relation**
- ▶ $I \subseteq S$ is the **set of initial states**
- ▶ AP is the set of **atomic propositions**
- ▶ $L : S \rightarrow 2^{AP}$ is the **labeling function**

Traces of Transition Systems

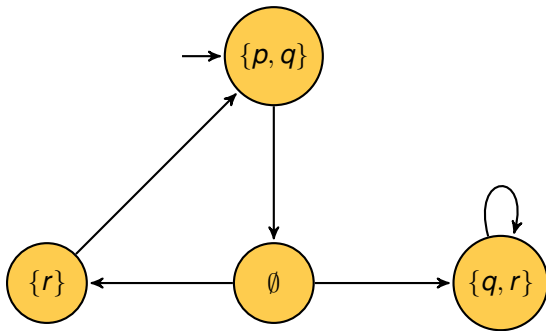
- ▶ Labels of the locations represent values of all observable propositions $\in AP$
- ▶ Captures system state
- ▶ Focus on sequences $L(s_0)L(s_1)\dots$ of labels of locations
- ▶ Such sequences are called **traces**
- ▶ Assuming transition systems have no terminal states,
 - ▶ Traces are infinite words over 2^{AP}
 - ▶ Traces $\in (2^{AP})^\omega$
 - ▶ Go to the example slide and define traces

Traces of Transition Systems

Given a transition system $TS = (S, Act, \rightarrow, I, AP, L)$ without terminal states,

- ▶ All maximal executions/paths are infinite
- ▶ Path $\pi = s_0 s_1 s_2 \dots$, $trace(\pi) = L(s_0)L(s_1)\dots$
- ▶ For a set Π of paths, $Trace(\Pi) = \{trace(\pi) \mid \pi \in \Pi\}$
- ▶ For a location s , $Traces(s) = Trace(Paths(s))$
- ▶ $Traces(TS) = \bigcup_{s \in I} Traces(s)$

Example Traces



$AP = \{p, q, r, t\}$

- ▶ $\{p, q\} \emptyset \{q, r\}^\omega$
- ▶ $(\{p, q\} \emptyset \{r\})^\omega$
- ▶ $(\{p, q\} \emptyset \{r\})^* \{p, q\} \emptyset \{q, r\}^\omega$

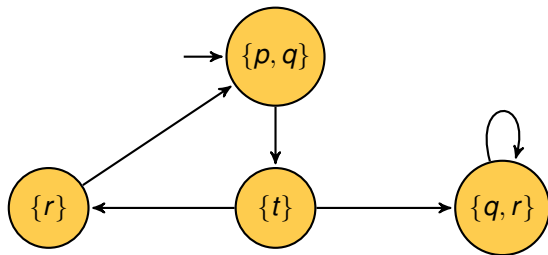
Linear Time Properties

- ▶ Linear-time properties specify traces that a TS must have
- ▶ A LT property P over AP is a subset of $(2^{AP})^\omega$
- ▶ TS over AP satisfies a LT property P over AP

$$TS \models P \text{ iff } \text{Traces}(TS) \subseteq P$$

- ▶ $s \in S$ satisfies LT property P (denoted $s \models P$) iff $\text{Traces}(s) \subseteq P$

Specifying Traces



- ▶ Whenever p is true, r will eventually become true
 - ▶ $\{A_0A_1A_2\cdots \mid \forall i \geq 0, p \in A_i \rightarrow \exists j \geq i, r \in A_j\}$
- ▶ q is true infinitely often
 - ▶ $\{A_0A_1A_2\cdots \mid \forall i \geq 0, \exists j \geq i, q \in A_j\}$
- ▶ Whenever r is true, so is q
 - ▶ $\{A_0A_1\cdots \mid \forall i \geq 0, r \in A_i \rightarrow q \in A_i\}$

Syntax of Linear Temporal Logic

Given AP , a set of propositions,

- ▶ Propositional logic formulae over AP
 - ▶ $a \in AP$ (atomic propositions)
 - ▶ $\neg\varphi, \varphi \wedge \psi, \varphi \vee \psi$
- ▶ Temporal Operators
 - ▶ $\bigcirc\varphi$ (Next φ)
 - ▶ $\varphi \mathbf{U}\psi$ (φ holds until a ψ -state is reached)
- ▶ LTL : Logic for describing LT properties

Semantics (On the board)

LTL formulae φ over AP interpreted over words $w \in \Sigma^\omega$, $\Sigma = 2^{AP}$,
 $w \models \varphi$

Derived Operators

- ▶ $true = \varphi \vee \neg\varphi$
- ▶ $false = \neg true$
- ▶ $\diamond\varphi = true \text{ U } \varphi$ (Eventually φ)
- ▶ $\Box\varphi = \neg\diamond\neg\varphi$ (Forever φ)

Precedence

- ▶ Unary Operators bind stronger than Binary
- ▶ \bigcirc and \neg equally strong
- ▶ U takes precedence over $\wedge, \vee, \rightarrow$
 - ▶ $a \vee b \text{ U } c \equiv a \vee (b \text{ U } c)$
 - ▶ $\bigcirc a \text{ U } \neg b \equiv (\bigcirc a) \text{ U } (\neg b)$

Examples

- ▶ Whenever the traffic light is red, it cannot become green immediately:
 $\Box(\text{red} \rightarrow \neg \bigcirc \text{green})$
- ▶ Eventually the traffic light will become yellow
 $\Diamond \text{yellow}$
- ▶ Once the traffic light becomes yellow, it will eventually become green
 $\Box(\text{yellow} \rightarrow \Diamond \text{green})$

Semantics over Infinite Words

Given LTL formula φ over AP ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

Let $\sigma = A_0 A_1 A_2 \dots$.

- ▶ $\sigma \models a$ iff $a \in A_0$
- ▶ $\sigma \models \varphi_1 \wedge \varphi_2$ iff $\sigma \models \varphi_1$ and $\sigma \models \varphi_2$
- ▶ $\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$
- ▶ $\sigma \models \bigcirc\varphi$ iff $A_1 A_2 \dots \models \varphi$
- ▶ $\sigma \models \varphi \mathbf{U} \psi$ iff
 $\exists j \geq 0$ such that $A_j A_{j+1} \dots \models \psi \wedge \forall 0 \leq i < j, A_i A_{i+1} \dots \models \varphi$

Semantics over Infinite Words

Given LTL formula φ over AP ,

$$L(\varphi) = \{\sigma \in (2^{AP})^\omega \mid \sigma \models \varphi\}$$

- ▶ $\sigma \models \Diamond \varphi$ iff $\exists j \geq 0, A_j A_{j+1} \dots \models \varphi$
- ▶ $\sigma \models \Box \varphi$ iff $\forall j \geq 0, A_j A_{j+1} \dots \models \varphi$
- ▶ $\sigma \models \Box \Diamond \varphi$ iff $\forall j \geq 0, \exists i \geq j, A_i A_{i+1} \dots \models \varphi$
- ▶ $\sigma \models \Diamond \Box \varphi$ iff $\exists j \geq 0, \forall i \geq j, A_i A_{i+1} \dots \models \varphi$

If $\sigma = A_0 A_1 A_2 \dots$, $\sigma \models \varphi$ is also written as $\sigma, 0 \models \varphi$. This simply means $A_0 A_1 A_2 \dots \models \varphi$. One can also define $\sigma, i \models \varphi$ to mean $A_i A_{i+1} A_{i+2} \dots \models \varphi$ to talk about a suffix of the word σ satisfying a property.

Transition System Semantics $TS \models \varphi$

Let $TS = (S, S_0, \rightarrow, AP, L)$ be a transition system, and φ an LTL formula over AP

- ▶ For an infinite path fragment π of TS ,

$$\pi \models \varphi \text{ iff } \text{trace}(\pi) \models \varphi$$

- ▶ For $s \in S$,

$$s \models \varphi \text{ iff } \forall \pi \in \text{Paths}(s), \pi \models \varphi$$

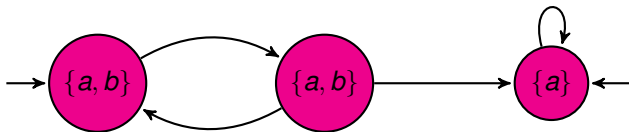
- ▶ $TS \models \varphi$ iff $\text{Traces}(TS) \subseteq L(\varphi)$

Transition System Semantics $TS \models \varphi$

Assume all states in TS are reachable from S_0 .

- ▶ $TS \models \varphi$ iff $TS \models L(\varphi)$ iff $Traces(TS) \subseteq L(\varphi)$
- ▶ $TS \models L(\varphi)$ iff $\pi \models \varphi \ \forall \pi \in Paths(TS)$
- ▶ $\pi \models \varphi \ \forall \pi \in Paths(TS)$ iff $s_0 \models \varphi \ \forall s_0 \in S_0$

Example

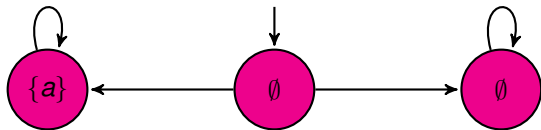


- ▶ $TS \models \Box a$,
- ▶ $TS \not\models \bigcirc(a \wedge b)$
- ▶ $TS \not\models (b \cup (a \wedge \neg b))$
- ▶ $TS \models \Box(\neg b \rightarrow \Box(a \wedge \neg b))$

More Semantics

- ▶ For paths π , $\pi \models \varphi$ iff $\pi \not\models \neg\varphi$
 $trace(\pi) \in L(\varphi)$ iff $trace(\pi) \notin L(\neg\varphi) = \overline{L(\varphi)}$
- ▶ $TS \not\models \varphi$ iff $TS \models \neg\varphi$?
 - ▶ $TS \models \neg\varphi \rightarrow \forall$ paths π of TS , $\pi \models \neg\varphi$
 - ▶ Thus, $\forall\pi$, $\pi \not\models \varphi$. Hence, $TS \not\models \varphi$
 - ▶ Now assume $TS \not\models \varphi$
 - ▶ Then \exists some path π in TS such that $\pi \models \neg\varphi$
 - ▶ However, there could be another path π' such that $\pi' \models \varphi$
 - ▶ Then $TS \not\models \neg\varphi$ as well
- ▶ Thus, $TS \not\models \varphi \not\equiv TS \models \neg\varphi$.

An Example



$TS \not\models \Diamond a$ and $TS \not\models \Box \neg a$

Equivalence of LTL Formulae

Equivalence

φ and ψ are equivalent ($\varphi \equiv \psi$) iff $L(\varphi) = L(\psi)$.

Expansion Laws

- ▶ $\varphi \text{ U } \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \text{ U } \psi))$
- ▶ $\diamond\varphi \equiv \varphi \vee \bigcirc\diamond\varphi$
- ▶ $\Box\varphi \equiv \varphi \wedge \bigcirc\Box\varphi$

Equivalence of LTL Formulae

φ and ψ are equivalent iff $L(\varphi) = L(\psi)$.

Equivalence of LTL Formulae

φ and ψ are equivalent iff $L(\varphi) = L(\psi)$.

Distribution

$$\bigcirc(\varphi \vee \psi) \equiv \bigcirc\varphi \vee \bigcirc\psi,$$

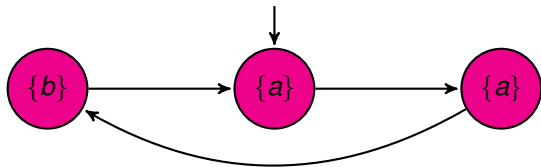
$$\bigcirc(\varphi \wedge \psi) \equiv \bigcirc\varphi \wedge \bigcirc\psi,$$

$$\bigcirc(\varphi \mathbf{U} \psi) \equiv (\bigcirc\varphi) \mathbf{U} (\bigcirc\psi),$$

$$\Diamond(\varphi \vee \psi) \equiv \Diamond\varphi \vee \Diamond\psi,$$

$$\Box(\varphi \wedge \psi) \equiv \Box\varphi \wedge \Box\psi$$

Equivalence of LTL Formulae



$$TS \models \Diamond a \wedge \Diamond b, TS \not\models \Diamond(a \wedge b)$$

$$TS \models \Box(a \vee b), TS \not\models \Box a \vee \Box b$$

Satisfiability, Model Checking of LTL

Two Questions

Given transition system TS , and an LTL formula φ . Does $TS \models \varphi$?

Given an LTL formula φ , is $L(\varphi) = \emptyset$?

How we go about this:

- ▶ Translate φ into an automaton A_φ that accepts infinite words such that $L(A_\varphi) = L(\varphi)$.
- ▶ Check for emptiness of A_φ to check satisfiability of φ .
- ▶ Check if $TS \cap \overline{A_\varphi}$ is empty, to answer the model-checking problem.

ω -automata

An ω -automaton is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$ where

- ▶ Q is a finite set of states
- ▶ Σ is a finite alphabet
- ▶ $\delta : Q \times \Sigma \rightarrow 2^Q$ is a state transition function (if non-deterministic, otherwise, $\delta : Q \times \Sigma \rightarrow Q$)
- ▶ $q_0 \in Q$ is an initial state and Acc is an acceptance condition

Run

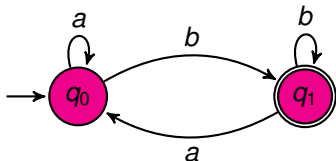
A run ρ of \mathcal{A} on an ω -word $\alpha = a_1 a_2 \dots \in \Sigma^\omega$ is an infinite state sequence $\rho(0)\rho(1)\rho(2)\dots$ such that

- ▶ $\rho(0) = q_0$,
- ▶ $\rho(i) = \delta(\rho(i-1), a_i)$ if \mathcal{A} is deterministic,
- ▶ $\rho(i) \in \delta(\rho(i-1), a_i)$ if \mathcal{A} is non-deterministic,

Büchi Acceptance

For Büchi Acceptance, Acc is specified as a set of states, $G \subseteq Q$. The ω -word α is accepted if there is a run ρ of α such that $Inf(\rho) \cap G \neq \emptyset$.

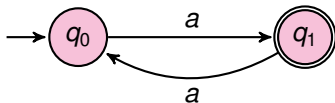
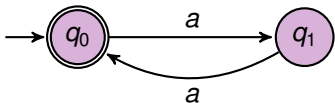
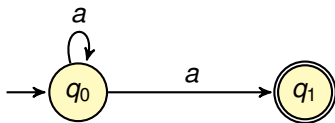
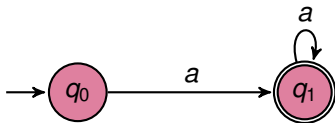
ω -Automata with Büchi Acceptance



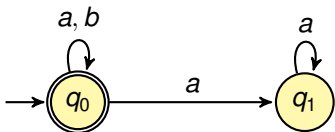
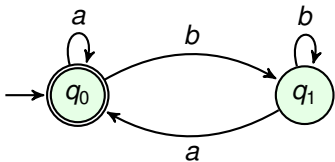
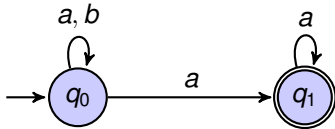
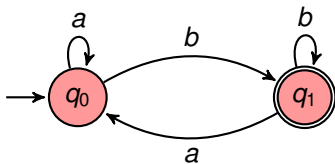
$$L(\mathcal{A}) = \{\alpha \in \Sigma^\omega \mid \alpha \text{ has a run } \rho \text{ such that } \text{Inf}(\rho) \cap G \neq \emptyset\}$$

Language accepted=Infinitely many b 's.

Comparing NFA and NBA



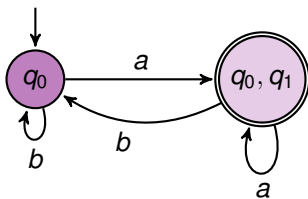
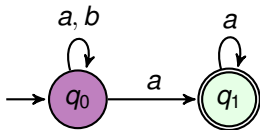
ω -Automata with Büchi Acceptance



- ▶ Left (T-B): Inf many b 's, Inf many a 's
- ▶ Right (T-B): Finitely many b 's, $(a + b)^\omega$

NBA and DBA

- ▶ Is every DBA as expressible as a NBA, like in the case of DFA and NFA?
- ▶ Can we do subset construction on NBA and obtain DBA?



NBA and DBA

There does not exist a deterministic Büchi automata capturing the language finitely many a 's.

