Hoare Logic and Model Checking

Model Checking
Lecture 8: Linear temporal logic (LTL)

Dominic Mulligan
Based on previous slides by Alan Mycroft and Mike Gordon

Programming, Logic, and Semantics Group
University of Cambridge

Academic year 2016–2017

Learning outcomes

By the end of this lecture, you should:

- Be familiar with the linear model of time
- Be familiar with LTL syntax and semantics

Linear model of time

Linear time

LTL's conception of time:

- At each moment in time exactly one successor state
- No "branching" of time into multiple futures
- Examine single execution of a system, and view it holistically

LTL can express **path properties** of systems

LTL formulae describe **infinite paths** through transition system

Right-serial transition systems

Suppose $\mathcal{T} = \langle S, S_0, \rightarrow \rangle$ is a transition system

Call $\mathcal{T}$ **right-serial** when:

for every $s \in S$ there exists $s' \in S$ such that $s \rightarrow s'$

Intuitively: “every state in S has a $\rightarrow$-successor”

Can convert any TS into right-serial TS:

- Add fresh state s to S
- Add transition $s \rightarrow s$ and $s' \rightarrow s$ for all terminal $s' \in S$

LTL syntax

Atomic propositions

Throughout we fix a set of **atomic propositions**, AP

Domain specific, and depend on modelling task

Recall examples from Lecture 1:

`lift_empty` `moving` `dispense`

other examples:

`cargo_bay_full` `student_in_lecture_theatre`

Will use p, q, r, and so on, to range over atomic propositions

LTL formulae

Define **LTL formulae** with the recursive grammar:

$$\begin{aligned} \phi, \psi, \xi, \ldots \quad & ::= \top \mid \bot \mid p \\ & ::= \neg \phi \\ & ::= \phi \wedge \psi \mid \phi \vee \psi \mid \phi \Rightarrow \psi \\ & ::= \Box \phi \mid \Diamond \phi \mid \bigcirc \phi \mid \phi \texttt{ UNTIL } \psi \end{aligned}$$

Intuitive explanation of LTL formulae

First line:

$$\top \mid \bot \mid p$$

$\top$, $\bot$, and $p \in AP$ are all LTL formulae

- $\top$ is the **logical truth** constant (or “true”),
- $\bot$ is the **logical falsity** constant (or “false”),
- p is embedding of **atomic propositions** into formulae

Intuitive explanation of LTL formulae

Second line:

$$\neg\phi$$

If ϕ is an LTL formula, then $\neg\phi$ is a formula

- $\neg\phi$ is **negation** (or "not ϕ")

Intuitive explanation of LTL formulae

Third line:

$$\phi \wedge \psi \mid \phi \vee \psi \mid \phi \Rightarrow \psi$$

If ϕ and ψ are LTL formulae, then so are $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \Rightarrow \psi$

- $\phi \wedge \psi$ is **conjunction** (or "ϕ and ψ")
- $\phi \vee \psi$ is **disjunction** (or "ϕ or ψ")
- $\phi \Rightarrow \psi$ is **implication** (or "if ϕ then ψ", or "ψ whenever ϕ")

Intuitive explanation of LTL formulae

Last line:

$$\Box\phi \mid \Diamond\phi \mid \bigcirc\phi \mid \phi \ \texttt{UNTIL} \ \psi$$

If ϕ and ψ are LTL formulae, then so are $\Box\phi$, $\Diamond\phi$, $\bigcirc\phi$ and $\phi \ \texttt{UNTIL} \ \psi$

- $\Box\phi$ is "henceforth ϕ", or "from now, always ϕ"
- $\Diamond\phi$ is "at some future point ϕ"
- $\bigcirc\phi$ is " immediately after ϕ", or " in the next state ϕ"
- $\phi \ \texttt{UNTIL} \ \psi$ is "at some future point ψ, but until then ϕ"

Alternative syntax for modalities

You may also see (e.g. in "Logic in Computer Science"):

- $G\phi$ instead of $\Box\phi$
- $F\phi$ instead of $\Diamond\phi$
- $X\phi$ instead of $\bigcirc\phi$

Author's preference, just syntactic differences

G = (G)lobally, F = (F)uture, X = Ne(X)t

Operator precedence

We add parentheses freely to disambiguate

Assign operator precedence to reduce number of parentheses:

- Unary $\neg$, $\Box$, $\Diamond$, and $\bigcirc$ bind most tightly
- After that `UNTIL`
- After that $\vee$ and $\wedge$
- Finally $\Rightarrow$ binds least tightly

Precedence examples

So:

$$\Box\phi \Rightarrow \Diamond \bigcirc \psi \quad \text{means} \quad (\Box\phi) \Rightarrow (\Diamond(\bigcirc\psi))$$

$$\phi \Rightarrow \psi \vee \Box\psi \quad \text{means} \quad \phi \Rightarrow (\phi \vee (\Box\psi))$$

$$\phi \vee \xi \Rightarrow \psi \; \texttt{UNTIL} \; \xi \quad \text{means} \quad (\phi \vee \xi) \Rightarrow (\psi \; \texttt{UNTIL} \; \xi)$$

and so on...

Example LTL formulae

Suppose `started` and `ready` are atomic propositions, then:

$$\Box\neg(\texttt{started} \wedge \neg\texttt{ready})$$

can be read as:

it is always the case that the system is never in a "started" state whilst not being "ready"

Example LTL formulae

Suppose `requested` and `acknowledged` are atomic propositions, then:

$$\Box(\texttt{requested} \Rightarrow \Diamond\texttt{acknowledged})$$

can be read as:

> *it is always the case that a "request" is always eventually "acknowledged" by the system*

Example LTL formulae

Suppose `enabled` is an atomic proposition, then:

$$\Box\Diamond\texttt{enabled}$$

can be read as:

it is always the case that the system is eventually "enabled"

the system is "enabled" infinitely often

Example LTL formulae

Suppose `deadlock` is an atomic proposition, then:

$$\Diamond\Box\texttt{deadlock}$$

can be read as:

eventually it will be always the case that the system is in "deadlock"

"deadlock" is inevitable

Semantics of LTL

Making intuition precise

Previous examples:

- Showed examples of properties expressible in LTL,
- Provided intuition for meaning of LTL formulae

Time to make that intuition precise...

Models for LTL

Suppose $\mathcal{T} = \langle S, S_0, \rightarrow \rangle$ is a **right-serial transition system**

Suppose $\mathcal{L} : S \longrightarrow \mathbb{P}(AP)$ is a **labelling function** for $\mathcal{T}$

Recall:

- Assigns sets of atomic propositions to states
- Intuitively: "which atomic propositions are true at a state"

Call $\langle S, S_0, \rightarrow, \mathcal{L} \rangle$ an **LTL model** (or just a **model**)

We use $\mathcal{M}, \mathcal{M}'$, and so on, to range over models

Infinite paths of states

A **path** in model $\mathcal{M}$ is an infinite sequence of states

$$s_0, s_1, s_2, s_3, \text{ and so on}$$

such that $s_i \to s_{i+1}$ and $s_i \in S$ for all i

Will also write $s_0 \to s_1 \to \ldots$ for a path

Will use π, π', and so on, to range over paths

Suffixes of paths

Suppose $\pi = s_0 \to s_1 \to s_2 \to \ldots$ is a path in some model

We write π^i to denote the i^{th} **suffix** of π

So π^2 is $s_2 \to s_3 \to \ldots$

And trivially $\pi^0 = \pi$

The suffix operation π^i just “chops off” i states from start of π

Path indexes

Suppose $\pi = s_0 \to s_1 \to s_2 \to \ldots$ is a path in some model

We write $\pi[i]$ to denote the i^{th} **index** of π

So $\pi[0] = s_0$ and $\pi[2] = s_2$ and $\pi[4] = s_4$, and so on

Satisfaction along a path

Suppose $\mathcal{M}$ is a model, π is a path in $\mathcal{M}$, and ϕ is an LTL formula

Define the **satisfaction relation** $\pi \models \phi$ recursively by:

$$
\begin{aligned}
\pi &\models \top && \text{always} \\
\pi &\models \bot && \text{never} \\
\pi &\models p && \text{iff } p \in \mathcal{L}(\pi[0]) \\
\pi &\models \neg\phi && \text{iff not } \pi \models \phi
\end{aligned}
$$

Satisfaction along a path

$$\begin{aligned}
\pi \models \phi \vee \psi & \quad \text{iff } \pi \models \phi \text{ or } \pi \models \psi \\
\pi \models \phi \wedge \psi & \quad \text{iff } \pi \models \phi \text{ and } \pi \models \psi \\
\pi \models \phi \Rightarrow \psi & \quad \text{iff not } \pi \models \phi \text{ or if } \pi \models \phi \text{ and } \pi \models \psi
\end{aligned}$$

Satisfaction along a path

$$
\begin{aligned}
\pi &\models \Box\phi && \text{iff } \pi^i \models \phi \text{ for all } i \geq 0 \\
\pi &\models \Diamond\phi && \text{iff } \pi^i \models \phi \text{ for some } i \geq 0 \\
\pi &\models \bigcirc\phi && \text{iff } \pi^1 \models \phi \\
\pi &\models \phi \;\texttt{UNTIL}\; \psi && \text{iff } \pi^i \models \psi \text{ for some } i \geq 0 \text{ and } \pi^j \models \phi \text{ for } 0 \leq j < i
\end{aligned}
$$

Notes on satisfaction relation

May also write $\mathcal{M}, \pi \models \phi$ to make model explicit

Note in clauses for $\Box\phi$ and $\Diamond\phi$:

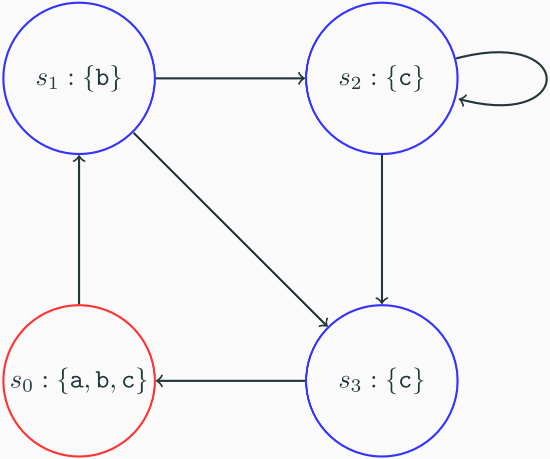
- Current state is counted as "future" too
- Makes some desirable properties hold of $\Box$ and $\Diamond$
- Matter of taste: some version of LTL do not permit this

$\Box\phi$ takes all suffixes of path π, whereas $\Diamond\phi$ takes some suffix

Note complexity of until—existential and universal quantification!

Examples

LTL model as a picture:



Examples

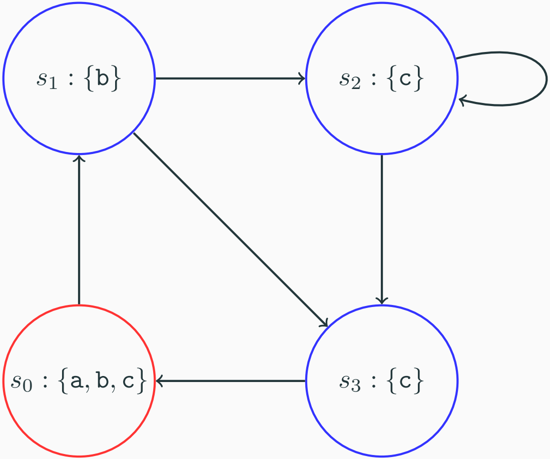
Consider path $\pi = s_0, s_1, s_2, s_2, s_2, \ldots$

Then:

- $\pi \models a$
- $\pi \models b \land \bigcirc b$
- $\pi \models c \Rightarrow \Diamond\, c$
- $\pi \models \Diamond\Box\, c$
- $\pi \models \neg\Box\Diamond\, a$

Examples

(Same model as before):



Examples

Consider path $\pi = s_0, s_1, s_2, s_3, s_0, s_1, s_2, s_3, \ldots$

Then:

- $\pi \models \Box\Diamond\, (a \wedge b \wedge c)$
- $\pi \models (\Box\Diamond\, a) \wedge (\Box\Diamond\, b) \wedge (\Box\Diamond\, c)$
- $\pi \models \Diamond(c \wedge \neg\, b)$
- $\pi \models \Box\Diamond\, a$

The LTL model checking problem

Model checking LTL

Given $\mathcal{M}$ and LTL formula ϕ

Establish whether $\mathcal{M}, \pi \models \phi$ for all π starting in initial states of $\mathcal{M}$

This is known as the **LTL model checking problem**

How do model checkers solve the LTL model checking problem?

A clue

Define (for some model $\mathcal{M}$):

$$Words(\phi) = \{\mathcal{L}(\pi) \mid \mathcal{M}, \pi \models \phi\}$$

$$Words(\mathcal{M}) = \{\mathcal{L}(\pi) \mid \text{for all } \pi \text{ in } \mathcal{M} \text{ s.t. } \pi[0] \in S_0\}$$

Here, $\mathcal{L}(\pi)$ is "mapping" of labelling function across states of π

As name suggests can be thought of as a set of "words":

- Alphabet is $\mathbb{P}(AP)$,
- Words are infinite, not finite, like regular words,
- Write $\mathbb{P}(AP)^\omega$ for set of all infinite words over $\mathbb{P}(AP)$,
- Note $Words(\phi) \subseteq \mathbb{P}(AP)^\omega$ and $Words(\mathcal{M}) \subseteq \mathbb{P}(AP)^\omega$

LTL model checking can be seen as a **language problem**, and natural tools to use are **automata**

A derivation

Note:

$$\begin{aligned}\mathcal{M} \models \phi \quad & \text{iff} \quad Words(\mathcal{M}) \subseteq Words(\phi) \\ & \text{iff} \quad Words(\mathcal{M}) \cap (\mathbb{P}(AP)^\omega \setminus Words(\phi)) = \{\} \\ & \text{iff} \quad Words(\mathcal{M}) \cap Words(\neg\phi) = \{\}\end{aligned}$$

$Words(\mathcal{M}) \subseteq Words(\phi)$ expresses that "all possible behaviours in $\mathcal{M}$ satisfy ϕ"

Also: recall $S \subseteq T$ iff $S \cap \bar{T} = \{\}$

Strategy

Suppose we have some automaton $A_{\neg\phi}$ that accepts infinite words, such that language of $A_{\neg\phi}$ is $Words(\neg\phi)$

Then combine to obtain an automaton $A_{\neg\phi} \otimes \mathcal{M}$

Constructed so that language of $A_{\neg\phi} \otimes \mathcal{M}$ is $Words(\neg\phi) \cap Words(\mathcal{M})$

Check for emptiness:

- If there is some word $w \in Words(\neg\phi) \cap Words(\mathcal{M})$ then this corresponds to a path π where $\mathcal{M}, \pi \models \neg\phi$
- Need to check if this path π starts in an initial state of $\mathcal{M}$
- If so, it is a counterexample to $\mathcal{M} \models \phi$, and therefore $\mathcal{M} \not\models \phi$,
- If no such path exists then $\mathcal{M} \models \phi$.

Some notes

This is a sketch:

- Not enough time to go into details of algorithm, as construction of automaton from LTL formulae ϕ fiddly,
- Logic in Computer Science avoids construction, see e.g. $\S 5.2$ of “Principles of model checking” for more details,
- Libraries exist for constructing automata from LTL formulae.

Need to use special type of automata: ω-automata, accept ω-regular languages, an infinite generalisation of regular languages

Common type to use for LTL model checking is Büchi automata, technique due to Vardi and Wolper

Using this technique, complexity of model checking is $O(V \cdot 2^{|\phi|})$

Summary

- LTL uses a linear model of time
- LTL formulae express “path properties” of systems
- LTL semantics with respect to infinite paths in model
- Can use automata-theoretic techniques to solve LTL model checking problem
- Complexity of LTL model checking exponential in size of formula