Formal Specification and Verification

Temporal logic (Part 2)

18.06.2013

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Temporal logic

Last time: Motivation

Linear Time Logic

Syntax

Π set of propositional variables.

The set of LTL (linear time logic) formulae is the smallest set such that:

- $\bot, \top$ and each propositional variable $P \in \Pi$ are formulae;
- if F, G are formulae, then so are $F \wedge G, F \vee G, \neg F$;
- if F, G are formulae, then so are $\bigcirc F$ and $F \mathcal{U} G$

Linear Time Logic

Syntax

Π set of propositional variables.

The set of LTL (linear time logic) formulae is the smallest set such that:

- each propositional letter $P \in \Pi$ is a formula;
- if F, G are formulae, then so are $F \wedge G, F \vee G, \neg F$;
- if F, G are formulae, then so are $\bigcirc F$ and $F \mathcal{U} G$

Remark: Instead of $\bigcirc F$ in some books also XF is used.

Linear Time Logic

Semantics

- Transition systems $(S, \rightarrow, L)$
  (with the property that for every $s \in S$ there exists $s' \in S$ with $s \rightarrow s'$
  i.e. no state of the system can "deadlock"[a])

  Transition systems are also simply called models in what follows.

[a] This is a technical convenience, and in fact it does not represent any real restriction on the systems we can model. If a system did deadlock, we could always add an extra state s_d representing deadlock, together with new transitions $s \rightarrow s_d$ for each s which was a deadlock in the old system, as well as $s_d \rightarrow s_d$.

Linear Time Logic

Semantics

- Transition systems $(S, \rightarrow, L)$
  (with the property that for every $s \in S$ there exists $s' \in S$ with $s \rightarrow s'$
  i.e. no state of the system can "deadlock"[a])

  Transition systems are also simply called models in what follows.

- Computation (execution, path) in a model $(S, \rightarrow, L)$
  infinite sequence of states $\pi = s_0, s_1, s_2, \ldots$ in S such that for each $i \geq 0$, $s_i \rightarrow s_{i+1}$.
  We write the path as $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$.

[a] This is a technical convenience, and in fact it does not represent any real restriction on the systems we can model. If a system did deadlock, we could always add an extra state s_d representing deadlock, together with new transitions $s \rightarrow s_d$ for each s which was a deadlock in the old system, as well as $s_d \rightarrow s_d$.

Linear Time Logic

Consider the path $\pi = s_0 \to s_1 \to \ldots.$

It represents a possible future of our system.

We write π^i for the suffix starting at s_i, e.g.,

$$\pi^3 = s_3 \to s_4 \to \ldots.$$

Linear Time Logic

Semantics

Let $TS = (S, \rightarrow, L)$ be a model and $\pi = s_0 \rightarrow \ldots$ be a path in TS.

Whether π satisfies an LTL formula is defined by the satisfaction relation $\models$ as follows:

- $\pi \models \top$
- $\pi \not\models \bot$
- $\pi \models p$ iff $p \in L(s_0)$, if $p \in \Pi$
- $\pi \models \neg F$ iff $\pi \not\models F$
- $\pi \models F \wedge G$ iff $\pi \models F$ and $\pi \models G$
- $\pi \models F \vee G$ iff $\pi \models F$ or $\pi \models G$
- $\pi \models \bigcirc F$ iff $\pi^1 \models F$
- $\pi \models F \mathcal{U} G$ iff $\exists m \geq 0$ $s.t.$ $\pi^m \models G$ and $\forall k \in \{0, \ldots, m-1\} : \pi^k \models F$

Linear Time Logic

Alternative way of defining the semantics:

An LTL structure M is an infinite sequence $S_0 S_1 \ldots$ with $S_i \subseteq \Pi$ for all $i \geq 0$. We define satisfaction of LTL formulas in M at time points $n \in \mathbb{N}$ as follows:

- $M, n \models p$ iff $p \in S_n$, if $p \in \Pi$
- $M, n \models F \wedge G$ iff $M, n \models F$ and $M, n \models G$
- $M, n \models F \vee G$ iff $M, n \models F$ or $M, n \models G$
- $M, n \models \neg F$ iff $M, n \not\models F$
- $M, n \models \bigcirc F$ iff $M, n+1 \models F$
- $M, n \models F \mathcal{U} G$ iff $\exists m \geq n$ *s.t.* $M, m \models G$ and $\forall k \in \{n, \ldots, m-1\} : M, k \models F$

Note that the time flow $(\mathbb{N}, <)$ is implicit.

Transition systems and LTL models

The connection between transition systems and LTL structures is as follows:

Every computation (evolution, path) of a transition system $s_0 \to s_1 \ldots$ gives rise to an LTL structure.

To see this, let $TS = (S, \to, L)$ be a transition system.

A computation $s_0, s_1, \ldots$ of TS induces an LTL structure $L(s_0)L(s_1)\ldots$

Such an LTL structure is called a trace of TS.

Abbreviations

- The future diamond

  $\Diamond\phi := \top\mathcal{U}\phi$

  $\pi \models \Diamond\phi$ iff $\exists m \geq 0 : \pi^m \models \phi$

- The future box

  $\Box\phi := \neg\Diamond\neg\phi$

  $\pi \models \Box\phi$ iff $\forall m \geq 0 : \pi^m \models \phi$

Abbreviations

- The future diamond

  $\Diamond\phi := \top\mathcal{U}\phi$ Sometimes denoted also $F\phi$

  $\pi \models \Diamond\phi$ iff $\exists m \geq 0 : \pi^m \models \phi$ $M, n \models \Diamond\phi$ iff $\exists m \geq n : M, m \models \phi$

- The future box

  $\Box\phi := \neg\Diamond\neg\phi$ Sometimes also denoted $G\phi$

  $\pi \models \Box\phi$ iff $\forall m{\geq}0 : \pi^m \models \phi$ $M, n \models \Box\phi$ iff $\forall m{\geq}n : M, m \models \phi$

Abbreviations

- The infinitely often operator

  $\Diamond^\infty \phi := \Box\Diamond\phi$

  $\pi \models \Diamond^\infty \phi$ iff $\{m \geq 0 \mid \pi^m \models \phi\}$ is infinite

  $M, n \models \Diamond^\infty \phi$ iff $\{m \geq n \mid M, m \models \phi\}$ is infinite

- The almost everywhere operator

  $\Box^\infty \phi := \Diamond\Box\phi$

  $\pi \models \Box^\infty \phi$ iff $\{m \geq 0 \mid \pi^m \not\models \phi\}$ is finite.

  $M, n \models \Box^\infty \phi$ iff $\{m \geq n \mid M, m \not\models \phi\}$ is finite.

Abbreviations

- The release operator
  $\phi \mathcal{R} \psi := \neg(\neg\phi \mathcal{U} \neg\psi)$

  $\pi \models \phi \mathcal{R} \psi$ iff $(\exists m \geq 0 : \pi^m \models \phi$ and $\forall k < m$: $\pi^k \models \psi)$ or
  $(\forall k \geq 0 : \pi^k \models \psi)$

  $M, n \models \phi \mathcal{R} \psi$ iff $(\exists m \geq n : M, m \models \phi$ and $\forall k < m : M, m \models \psi)$ or
  $(\forall k \geq m : M, k \models \psi)$

  Read as
  "ψ always holds unless released by ϕ" i.e.,
  "ψ holds permanently up to and including the first point where ϕ holds (such an ϕ-point need not exist at all)".

Abbreviations

- The strict until operator:
  $F\mathcal{U}^{<}G := \bigcirc(F\mathcal{U}G)$
  $M, n \models F\mathcal{U}^{<}G$ iff $\exists m > n : M, m \models G \wedge \forall k \in \{n+1, ..., m-1\}, M, k \models F$

The difference between standard and strict until is that strict until requires G to happen in the strict future and that F needs not hold true of the current point.

Equivalence

We say that two LTL formulas F and G are (globally) equivalent (written $F \equiv G$)
if, for all LTL structures M and $i \geq 0$, we have $M, i \models F$ iff $M, i \models G$.

Note that:

$\bigcirc F \equiv \bot \, \mathcal{U}^{<} F$ and

$F \mathcal{U} G \equiv G \vee (F \wedge (F \mathcal{U}^{<} G))$

Thus, an equally expressive version of LTL is obtained by using $\mathcal{U}^{<}$ as the only temporal operator.

This cannot be done with the standard until

Equivalence

Some useful equivalences that will be useful later on (exercise: prove them):

$\neg \bigcirc F \equiv \bigcirc \neg F$	(self-duality of next)
$\Diamond\Diamond F \equiv \Diamond F$	(idempotency of diamond)
$\bigcirc\Diamond F \equiv \Diamond \bigcirc F$	(commutation of next with Diamond)
$\Diamond\Diamond^{\infty} F \equiv \Diamond^{\infty} F \equiv \Diamond^{\infty}\Diamond F$	(absorption of diamonds by "infinitely often"
$F\mathcal{U}G \equiv \neg(\neg F\mathcal{R}\neg G)$	(until and release are duals)
$F\mathcal{U}G \equiv G \vee (F \wedge \bigcirc(F\mathcal{U}G))$	(unfolding of until)
$F\mathcal{R}G \equiv (F \wedge G) \vee (G \wedge \bigcirc(F\mathcal{R}G))$	(unfolding of release)

Temporal Properties

A temporal property is a set of LTL structures
(those on which the property is true).

Thus, a temporal property P can be defined using an LTL formula F:

$$P = \{M \mid M, 0 \models F\}.$$

When given a transition system TS representing a reactive system and an LTL formula F representing a temporal property,

TS satisfies F if $M, 0 \models F$ for all traces M of TS.

In this case, we write $TS \models F$.

Typical properties of reactive systems that need to be checked during verification are safety properties, liveness properties, and fairness properties.

Safety properties

Intuitively, a safety property asserts that "nothing bad happens"

general form: Condition $\rightarrow \Box F_{\text{Safe}}$

Examples of safety properties:

- **Mutual Exclusion.** For the example:

$$\Box(\neg((A = 2) \wedge (B = 2)))$$

- **Freedom from Deadlocks:** At any time, some process should be enabled:

$$\Box(enabled_1 \vee \cdots \vee enabled_k)$$

- **Partial Correctness:** If F is satisfied when the program starts, then G will be satisfied if the program reaches a distinguished state:

$$F \rightarrow \Box(\text{Dist} \rightarrow G)$$

where Dist $\in \Pi$ marks the distinguished state.

Liveness properties

Intuitively, a liveness property asserts that "something good will happen"

Examples of liveness properties:

- **Guaranteed Accessibility.** For the example:

$$\Box(A = 1 \rightarrow \Diamond(A = 2)) \wedge \Box(B = 1 \rightarrow \Diamond(B = 2))$$

- **Responsiveness:** If a request is issued, it will eventually be granted:

$$\Box(\text{req} \rightarrow \Diamond\text{grant})$$

- **Total Correctness:** If F is satisfied when the program starts, then the program terminates in a distinguished state where G is satisfied:

$$\phi \rightarrow \Diamond(\text{Dist} \wedge G)$$

  Note that, in contrast, partial correctness is a safety property.

Fairness properties

When modelling concurrent systems, it is usually important to make some fairness assumptions. Assume that there are k processes, that $\text{enabled}_i \in \Pi$ is true in a state s if process $\#i$ is enabled in s for execution, and that executed_i is true in a state s if process $\#i$ has been executed to reach s.

Examples of fairness properties

- **Unconditional Fairness:** Every process is executed infinitely often:

$$\bigwedge_{1 \leq i \leq k} \Diamond^{\infty} \text{executed}_i$$

  Unconditional fairness is appropriate when processes can (and should!) be executed and any time. This is not always the case.

Fairness properties

When modelling concurrent systems, it is usually important to make some fairness assumptions. Assume that there are k processes, that $\text{enabled}_i \in \Pi$ is true in a state s if process $\#i$ is enabled in s for execution, and that executed_i is true in a state s if process $\#i$ has been executed to reach s.

Examples of fairness properties

- **Strong Fairness:** Every process enabled infinitely often is executed infinitely often:

$$\bigwedge_{1 \leq i \leq k} (\Diamond^{\infty} \text{enabled}_i \rightarrow \Diamond^{\infty} \text{executed}_i)$$

  Processes enabled only finitely often need not be guaranteed to be executed: they eventually and forever retract being enabled.

Fairness properties

When modelling concurrent systems, it is usually important to make some fairness assumptions. Assume that there are k processes, that $\text{enabled}_i \in \Pi$ is true in a state s if process $\#i$ is enabled in s for execution, and that executed_i is true in a state s if process $\#i$ has been executed to reach s.

Examples of fairness properties

- **Weak Fairness:** Every process enabled almost everywhere is executed infinitely often.

$$\bigwedge_{1 \leq i \leq k} (\Box^{\infty} \text{enabled}_i \rightarrow \Diamond^{\infty} \text{executed}_i)$$

  This means that a process cannot be enabled constantly in an infinite interval without being executed in this interval.