Formal Specification and Verification

Temporal logic (2)

10.01.2017

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Last time

Motivation

Example

Transition systems

- computations
- computation trees

Transition systems

We use an abstract model of reactive and concurrent systems.

Definition (Transition system, simplified version)

Let Π be a finite set of propositional variables.

A transition system is a tuple $(S, \to, S_i, L)$ with

- S a non-empty set of states;
- $\to \subseteq S \times S$ is a transition relation that is total, i.e. for each state $s \in S$, there is a state $s' \in S$ such that $s \to s'$;
- $S_i \subseteq S$ is a set of initial states;
- $L : S \to \{0, 1\}^{AP}$ is a valuation function which we will also regard as a function $L : AP \times S \to \{0, 1\}$

Computations

Let $TS = (S, \rightarrow, S_i, L)$ be a transition system.

A computation (or execution) of TS is an infinite sequence $s_0 s_1 \ldots$ of states such that $s_0 \in S_i$ and $s_i \rightarrow s_{i+1}$ for all $i \geq 0$.

Computation trees

Transition systems can be non-deterministic, i.e., for an $s \in S$, the set $\{s' \mid s \to s'\}$ can have arbitrary cardinality > 0.

Thus, in general there is more than a single computation.

Instead of considering single computations in isolation, we can arrange all of them in a computation tree.

Informally, for $s \in S_i$, the (infinite) computation tree $T(TS, s)$ of TS at $s \in S$ is inductively constructed as follows:

- use s as the root node;
- for each leaf s' of the tree, add successors $\{t \in S \mid s' \to t\}$.

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:

- $\bot, \top$ 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 $F \mathcal{U} G$

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 $(S, \to, L)$
  (with the property that for every $s \in S$ there exists $s' \in S$ with $s \to s'$
  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 $(S, \to, L)$
  infinite sequence of states $\pi = s_0, s_1, s_2, \ldots$ in S such that for each $i \geq 0$, $s_i \to s_{i+1}$.
  We write the path as $s_0 \to s_1 \to s_2 \to \ldots$.

[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 $s \to s_d$ for each s which was a deadlock in the old system, as well as $s_d \to s_d$.

Linear Time Logic

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

It represents a possible future of our system.

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

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

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 $S_0S_1\ldots$ 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 $p \in S_n, \qquad$ 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 $s_0 \rightarrow s_1 \ldots$ gives rise to an LTL structure.

> To see this, let $TS = (S, \rightarrow, L)$ 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:
  $F\mathcal{U}^{<}G := \bigcirc(F\mathcal{U}G)$

  $\pi \models F\mathcal{U}^{<}G$ iff $\exists m > 0 : \pi^m \models G \wedge \forall k \in \{1, 2, \ldots, m-1\}, \pi^k \models F$

  $M, n \models F\mathcal{U}^{<}G$ 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 $M, i \models G$.

equivalently:

if for all transition systems T and all paths π in T we have:
$\pi \models F$ iff $\pi \models G$.

Note that:

$\bigcirc F \equiv \perp \mathcal{U}^{<} F$ 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 ofte
$F \mathcal{U} G \equiv \neg(\neg F \mathcal{R} \neg G)$	(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)