T-79.186 Reactive Systems

Spring 2003, Lecture 3

Keijo Heljanko

January 28, 2003

7 Temporal Logics

Temporal logics are currently the most widely used specification formalism for reactive systems. They were first suggested to be used for specifying properties of programs in late 1970's by A. Pnueli. Before that philosophers had used similar logics to reason about the notions of knowledge and belief in natural language. In early 1980's first practical tools using temporal logics as the specification language appeared.

Temporal logics proved to be popular for several reasons:

- specifications have close correspondence to the natural language
- well defined semantics
- algorithms and tools for them exist
- expressive enough
- are succinct enough (more succinct than automata)
- allow easy complementation without blow-up

Recall the definition of Kripke structures:

Definition 6.1 Let AP be a non-empty set of atomic propositions. A Kripke structure is a four tuple $M = (S, s^0, R, L)$, where

- S is a finite set of states,
- $s^0 \in S$ is an initial state,
- $R \subseteq S \times S$ is a transition relation, for which it holds that $\forall s \in S : \exists s' \in S : (s, s') \in R$, and
- $L : S \rightarrow 2^{AP}$ is labelling, a function which labels each state with the atomic propositions which hold in that state.

In temporal logics we are interested in non-terminating (infinite) executions of the system, whose behavior is described by the Kripke structure M.

A **path** starting from a state $s \in S$ is an infinite sequence of states $\pi = s_0, s_1, s_2, \ldots$, such that $s_0 = s$ and $(s_i, s_{i+1}) \in R$ holds for all $i \geq 0$.

7.1 Temporal Logic LTL

We start by defining the syntax of LTL, the logic we will mostly use in this course.

The logic LTL is a linear temporal logic, meaning that the formulas are interpreted over infinite sequences of states.

A minimal syntax for LTL formulas is the following. Given the set AP, an LTL formula is:

- $\mathbf{true}$, $\mathbf{false}$, or p for $p \in AP$.
- $\neg f_1, f_1 \vee f_2, X f_1$, or $f_1 \, U \, f_2$, where f_1 and f_2 are LTL formulas.

The formula $X f_1$ is read "next-time f_1", and $f_1 \, U \, f_2$ is read "f_1 until f_2".

We often a non-minimal version of the LTL syntax.

Given the set AP, an LTL formula is:

- **true**, **false**, or p for $p \in AP$.
- $\neg f_1, f_1 \vee f_2, f_1 \wedge f_2, X f_1, f_1 \, U \, f_2$, or $f_1 \, R \, f_2$, where f_1 and f_2 are LTL formulas.

The formula $f_1 \, R \, f_2$ is read "f_1 releases f_2".

This syntax is redundant, because actually $f_1 \wedge f_2 = \neg(\neg f_1 \vee \neg f_2)$, $f_1 \, R \, f_2 = \neg(\neg f_1 \, U \, \neg f_2)$ and $X f_1 = \neg X \neg f_1$ hold in all temporal logics to be considered in this course.

The redundancy is used by some algorithms to push negations deeper into the formula by using the temporal logic DeMorgan rules described above.

We also use standard propositional shorthands: $f_1 \Rightarrow f_2 = (\neg f_1 \lor f_2)$, $f_1 \Leftrightarrow f_2 = ((f_1 \land f_2) \lor (\neg f_1 \land \neg f_2))$, etc.

Some often used temporal logic shorthands are:

- $F f_1 = \mathbf{true}\, U\, f_1$,
  read "finally f_1" (or "in the future f_1")
- $G f_1 = \neg F \neg f_1$ (or equivalently $G f_1 = \mathbf{false}\, R\, f_1$),
  read "globally f_1" (or "always f_1").

Also also the following notation is used often with LTL: $\Box f_1 = G f_1$ (for "box f_1"), $\Diamond f_1 = F f_1$ (for "diamond f_1"), and $\bigcirc f_1 = X f_1$ (for "next-time f_1").

$GF f_1$ is often read as "infinitely often", and there are several less common shorthands in the literature.

Some examples of practical use of LTL formulas in specification are:

- $\Box\neg(cs_1 \wedge cs_2)$ (it always holds that two processes are not at the same time in a critical section),
- $\Box(req \rightarrow \Diamond ack)$ (it is always the case that a request is eventually followed by an acknowledgement), and
- $((\Box\Diamond sch_1) \wedge (\Box\Diamond sch_2)) \rightarrow (\Box(tr_1 \rightarrow \Diamond cs_1))$ (if both process 1 and 2 are scheduled infinitely often, then always the entering of process 1 in the trying section is followed by the process 1 eventually entering the critical section).

We can now define the **semantics** of LTL in a path.

We denote the fact that an LTL formula f holds in a path π with the notation $\pi \models f$.

Another way of saying $\pi \models f$ is that the path π is a **model** of the formula f.

The term **model checking** in fact refers to the fact that we are checking whether the behaviors of the system are models of the specification formula.

We use π^i to denote the suffix of the path $\pi = s_0, s_1, s_2, \ldots$ starting at index i, and thus $\pi^i = s_i, s_{i+1}, s_{i+2}, \ldots$.

We use π_0 to denote the first state of the path $\pi = s_0, s_1, s_2, \ldots$, namely s_0.

We next inductively define the $\models$ relation (when does a path π of the system fulfill its specification f).

The relation $\pi \models f$ is defined inductively as follows:

- $\pi \models \mathbf{true}$
- $\pi \not\models \mathbf{false}$
- $\pi \models p$ iff $p \in L(\pi_0)$ for $p \in AP$
- $\pi \models \neg f_1$ iff not $\pi \models f_1$
- $\pi \models f_1 \vee f_2$ iff $\pi \models f_1$ or $\pi \models f_2$
- $\pi \models f_1 \wedge f_2$ iff $\pi \models f_1$ and $\pi \models f_2$
- $\pi \models X\, f_1$ iff $\pi^1 \models f_1$
- $\pi \models f_1\, U\, f_2$ iff there exists $j \geq 0$, such that $\pi^j \models f_2$ and for all $0 \leq i < j$, $\pi^i \models f_1$
- $\pi \models f_1\, R\, f_2$ iff for all $j \geq 0$, if for every $0 \leq i < j$ $\pi^i \not\models f_1$ then $\pi^j \models f_2$.

We also want to define when a formula f holds in a state s in a Kripke structure M.

Definition 7.2 An LTL formula f holds in a state s of Kripke structure M, denoted $M, s \models f$, iff for all paths π in M, such that $\pi_0 = s$, it holds that $\pi \models f$.

Thus a formula f holds in s iff it holds for all paths starting at s.

Now we want to also define when a Kripke structure satisfies the LTL specification f.

Definition 7.3 An LTL formula f holds in a Kripke structure M, denoted $M \models f$ iff $M, s^0 \models f$.

We will in the future use logics which have both explicit A :"for all paths", and E : "there exist a path" quantifications in the language. The standard LTL interpretation adds an implicit "for all paths" quantification in front of (the top-level) LTL formula. Intuitively this means that out LTL specifications specify properties which should hold for all behaviors (paths) of the system.