Model Checking General Temporal Formulas

Next, we consider methods for model checking general LTL formulas.

Let $D$ be an FDS and $\varphi$ an LTL formula. Assume we wish to check whether $D \models \varphi$. We proceed along the following steps:

- Construct the temporal tester $T(\neg \varphi)$. This is an FDS whose computations are all the sequences falsifying $\varphi$.

- Form the parallel composition $D \parallel T(\neg \varphi)$. This is an FDS whose computations are all computations of $D$ which violate $\varphi$.

- Check whether the composition $D \parallel T(\neg \varphi)$ is feasible. $D \models \varphi$ iff $D \parallel T(\neg \varphi)$ is infeasible.

It only remains to describe the construction of a tester $T(\psi)$ for a general LTL formula $\psi$. 
Operations on FDS’s: Asynchronous Parallel Composition

The asynchronous parallel composition of systems $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted by $\mathcal{D}_1 \parallel \mathcal{D}_2$, is given by $\mathcal{D} = \langle V, \Theta, \rho, J, C \rangle$, where

\begin{align*}
V &= V_1 \cup V_2 \\
\Theta &= \Theta_1 \land \Theta_2 \\
\rho &= \left( \rho_1 \land \text{pres}(V_2 - V_1) \right) \lor \left( \rho_2 \land \text{pres}(V_1 - V_2) \right) \\
J &= J_1 \cup J_2 \\
C &= C_1 \cup C_2
\end{align*}

The predicate $\text{pres}(U)$ stands for the assertion $U' = U$, implying that all the variables in $U$ are preserved by the transition.

Asynchronous parallel composition represents the interleaving-based concurrency which is the assumed concurrency in shared-variables models.

**Claim 5.** $\mathcal{D}(P_1 \parallel P_2) \sim \mathcal{D}(P_1) \parallel \mathcal{D}(P_2)$

That is, the FDS corresponding to the program $P_1 \parallel P_2$ is equivalent to the asynchronous parallel composition of the FDS’s corresponding to $P_1$ and $P_2$. 
Synchronous Parallel Composition

The synchronous parallel composition of systems $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted by $\mathcal{D}_1 \ || \ \mathcal{D}_2$, is given by the FDS $\mathcal{D} = \langle V, \Theta, \rho, \mathcal{J}, C \rangle$, where

\[
\begin{align*}
V &= V_1 \cup V_2 \\
\Theta &= \Theta_1 \land \Theta_2 \\
\rho &= \rho_1 \land \rho_2 \\
\mathcal{J} &= \mathcal{J}_1 \cup \mathcal{J}_2 \\
C &= C_1 \cup C_2
\end{align*}
\]

Synchronous parallel composition can be used for hardware verification, where it is the natural operator for combining two circuits into a composed circuit. Here we use it for model checking of LTL formulas.

**Claim 6.** The sequence $\sigma$ of $V$-states is a computation of the combined $\mathcal{D}_1 \ || \ \mathcal{D}_2$ iff $\sigma \downarrow_{V_1}$ is a computation of $\mathcal{D}_1$ and $\sigma \downarrow_{V_2}$ is a computation of $\mathcal{D}_2$.

Here, $\sigma \downarrow_{V_i}$ denotes the sequence obtained from $\sigma$ by restricting each of the states to a $V_i$-state.
Temporal Testers

Let $\varphi$ be a temporal formula over vocabulary $U$, and let $x \notin U$ be a boolean variable disjoint from $U$.

In the following, let $\sigma : s_0, s_1, \ldots$ be an infinite sequence of states over $U \cup \{x\}$. We say that $x$ matches $\varphi$ in $\sigma$ if, for every position $j \geq 0$, the value of $x$ at position $j$ is true iff $(\sigma, j) \models \varphi$.

A temporal tester for $\varphi$ is an FDS $T(\varphi)$ over $U \cup \{x\}$, satisfying the requirement:

The infinite sequence $\sigma$ is a computation of $T(\varphi)$ iff $x$ matches $\varphi$ in $\sigma$.

A consequence of this definition is that every infinite sequence $\pi$ of $U$-states can be extended into a computation $\sigma$ of $T(\varphi)$ by interpreting $x$ at position $j \geq 0$ of $\sigma$ as $1$ iff $(\pi, j) \models \varphi$. 

Construction of Temporal Testers

A formula $\varphi$ is called a principally temporal formula (PTF) if the main operator of $p$ is temporal. A PTF is called a basic temporal formula if it contains no other PTF as a proper sub-formula.

We start our construction by presenting temporal testers for the basic temporal formulas.
A Tester for $\bigcirc p$

The tester for the formula $\bigcirc p$ is given by:

$$T(\bigcirc p) : \begin{cases} 
  V : & \text{Vars}(p) \cup \{x\} \\
  \Theta : & 1 \\
  \rho : & x = p' \\
  \mathcal{J} = C : & \emptyset 
\end{cases}$$

**Claim 7.**
$T(\bigcirc p)$ is a temporal tester for $\bigcirc p$.

**Proof:**
Let $\sigma$ be a computation of $T(\bigcirc p)$. We will show that $x$ matches $\bigcirc p$ in $\sigma$. Let $j \geq 0$ be any position. By the transition relation, $x = 1$ at position $j$ iff $s_{j+1} \models p$ iff $(\sigma, j) \models \bigcirc p$.

Let $\sigma$ be an infinite sequence such that $x$ matches $\bigcirc p$ in $\sigma$. We will show that $\sigma$ is a computation of $T(\bigcirc p)$. For any position $j \geq 0$, $x = 1$ at $j$ iff $(\sigma, j) \models \bigcirc p$, iff $s_{j+1} \models p$. Thus, $x$ satisfies $x = p'$ at every position $j$. 

}$
A Tester for $pUq$

The tester for the formula $pUq$ is given by:

$$T(pUq) : \begin{cases} V : & \text{Vars}(p, q) \cup \{x\} \\ \Theta : & 1 \\ \rho : & x = q \lor (p \land x') \\ \iota : & q \lor \neg x \\ C : & \emptyset \end{cases}$$

**Claim 8.**

$T(pUq)$ is a temporal tester for $pUq$.

**Proof:**

Let $\sigma$ be a computation of $T(pUq)$. We will show that $x$ matches $pUq$ in $\sigma$. Let $j \geq 0$ be any position. Consider first the case that $s_j \models x$ and we will show that $(\sigma, j) \models pUq$. According to the transition relation, $s_j \models x$ implies that either $s_j \models q$ or $s_j \models p$ and $s_{j+1} \models x$. If $s_j \models q$ then $(\sigma, j) \models pUq$ and we are done. Otherwise, we apply the same argument to position $j + 1$. Continuing in this manner, we either locate a $k \geq j$ such that $s_k \models q$ and $s_i \models p$ for all $i$, $j \leq i < k$, or we have $s_i \models \neg q \land p \land x$ for all $i \geq j$. If we locate a stopping $k$ then, obviously $(\sigma, j) \models pUq$ according to the semantic definition of the $U$ operator. The other case in which both $\neg q$ and $x$ hold over all positions beyond $j$ is impossible since it violates the justice requirement demanding that $\sigma$ contains infinitely many positions at which either $q$ is true or $x$ is false.
Proof Continued

Next we consider the case that $\sigma$ is a computation of $T(pUq)$ and $(\sigma,j) \models pUq$, and we have to show that $s_j \models x$. According to the semantic definition, there exists a $k \geq j$ such that $s_k \models q$ and $s_i \models p$ for all $i, j \leq i < k$. Proceeding from $k$ backwards all the way to $j$, we can show (by induction if necessary) that the transition relation implies that $s_t \models x$ for all $t = k, k - 1, \ldots, j$.

In the other direction, let $\sigma$ be an infinite sequence such that $x$ matches $pUq$ in $\sigma$. We will show that $\sigma$ is a computation of $T(pUq)$. From the semantic definition of $U$ it follows that $(\sigma,j) \models pUq$ iff either $s_j \models q$ or $s_j \models p$ and $(\sigma,j+1) \models pUq$. Thus, if $x = (pUq)$ at all positions, the transition relation $x = q \lor (p \land x')$ holds at all positions. To show that $x$ satisfies the justice requirement $q \lor \neg x$ it is enough to consider the case that $\sigma$ contains only finitely many $q$-positions. In that case, there must exist a cutoff position $c \geq 0$ such that no position beyond $c$ satisfies $q$. In this case, $pUq$ must be false at all positions beyond $c$. Consequently, $x$ is false at all positions beyond $c$ and is therefore false at infinitely many positions. \(\blacksquare\)
Why Do We Need the Justice Requirement

Reconsider the temporal tester for $p \mathcal{U} q$:

$$T(p \mathcal{U} q) : \begin{cases} V : & \text{Vars}(p, q) \cup \{x\} \\ \Theta : & 1 \\ \rho : & x = q \lor (p \land x') \\ \mathcal{J} : & q \lor \neg x \\ C : & \emptyset \end{cases}$$

We wish to show that the justice requirement $q \lor \neg x$ is essential for the correctness of the construction. Consider a state sequence $\sigma : s_0, s_1, \ldots$ in which $q$ is identically false and $p$ is identically true at all positions. In this case, the transition relation reduces to the equation

$$x = x'.$$

This equation has two possible solutions, one in which $x$ is identically false and the other in which $x$ is identically true at all positions. Only $x = 0$ matches $p \mathcal{U} q$. This is also the only solution which satisfies the justice requirement.

Thus, the role of the justice requirement is to select among several solutions to the transition relation equation, a unique one which matches the basic temporal formula at all positions.
A Tester for $p \mathcal{W} q$

A supporting evidence for the significance of the justice requirements is provided by the tester for the formula $p \mathcal{W} q$:

$$
T(p \mathcal{W} q) : \begin{cases}
V : & \text{Vars}(p, q) \cup \{x\} \\
\Theta : & 1 \\
\rho : & x = q \lor (p \land x') \\
J : & \neg p \lor x \\
C : & \emptyset
\end{cases}
$$

Note that the transition relation of $T(p \mathcal{W} q)$ is identical to that of $T(p \mathcal{U} q)$, and they only differ in their respective justice requirements.

The role of the justice requirement in $T(p \mathcal{W} q)$ is to eliminate the solution $x = 0$ over a computation in which $p = 1$ and $q = 0$ at all positions.
Testers for the Derived Operators

Based on the testers for $U$ and $W$, we can construct testers for the derived operators $\Diamond$ and $\Box$. They are given by

$$T(\Diamond p) : \begin{cases} V : & \text{Vars}(p) \cup \{x\} \\ \Theta : & 1 \\ \rho : & x = p \lor x' \\ J : & p \lor \neg x \\ C : & \emptyset \end{cases}$$

$$T(\Box p) : \begin{cases} V : & \text{Vars}(p) \cup \{x\} \\ \Theta : & 1 \\ \rho : & x = p \land x' \\ J : & \neg p \lor x \\ C : & \emptyset \end{cases}$$

A formula such as $\Diamond p$ can be viewed as a “promise for an eventual $p$”. The justice requirement $p \lor \neg x$ can be interpreted as suggesting:

**Either fulfill all your promises or stop promising.**

Note that once $x = 0$ in the tester $T(\Diamond p)$, it remains 0 and requires $p = 0$ ever after.
Testers for the Past Basic Formulas

The following are testers for the basic past formulas $\Box p$ and $pS q$:

\[
T(\Box p) : \begin{cases} 
V & : \text{Vars}(p) \cup \{x\} \\
\Theta & : x = 0 \\
\rho & : x' = p \\
\mathcal{J} & : \emptyset \\
C & : \emptyset 
\end{cases} \quad T(pS q) : \begin{cases} 
V & : \text{Vars}(p, q) \cup \{x\} \\
\Theta & : x = q \\
\rho & : x' = q' \lor (p' \land x) \\
\mathcal{J} & : \emptyset \\
C & : \emptyset 
\end{cases}
\]

Note that testers for past formulas are not associated with any fairness requirements. On the other hand, they have a non-trivial initial conditions.
Testers for Compound Temporal Formulas

Up to now we only considered testers for basic formulas. The construction for non-basic formulas is based on the following reduction principle. Let $f(\varphi)$ be a temporal formula containing one or more occurrences of the basic formula $\varphi$. Then the temporal tester for $f(\varphi)$ can be constructed according to the following recipe:

$$T(f(\varphi)) = T(f(x_\varphi)) \parallel T(\varphi)$$

where, $x_\varphi$ is the boolean output variable of $T(\varphi)$, and $f(x_\varphi)$ is obtained from $f(\varphi)$ by replacing every instance of $\varphi$ by $x_\varphi$.

Following this recipe the temporal tester for an arbitrary formula $f$ can be decomposed into a synchronous parallel composition of smaller testers, one for each basic formula nested within $f$. 
Example: A Tester for $\diamondsuit \Box p$

Following is a tester for the formula $\diamondsuit \Box p$ which is obtained by computing the parallel composition $T(\diamondsuit x_{\Box}) \parallel T(\Box p)$.

$$T(\diamondsuit \Box p) : \begin{cases} 
V : & \text{Vars}(p) \cup \{x_{\diamondsuit}, x_{\Box}\} \\
\Theta : & 1 \\
\rho : & (x_{\Box} = p \land x_{\Box}') \land (x_{\diamondsuit} = x_{\Box} \lor x_{\diamondsuit}') \\
\mathcal{J} : & \{\neg p \lor x_{\Box}, \ x_{\Box} \lor \neg x_{\diamondsuit}\} \\
C : & \emptyset 
\end{cases}$$

The output variable of $\diamondsuit \Box p$ is $x_{\diamondsuit}$.
Model Checking General Temporal Formulas

To check whether $\mathcal{D} \models \varphi$, perform the following steps:

- **Construct** the temporal tester $T(\varphi)$.

- **Form** the combined system $\mathcal{C} = \mathcal{D} \parallel T(\varphi) \parallel [\Theta : \neg x \varphi]$, where $[\Theta : \neg x \varphi]$ is a trivial FDS which imposes the initial condition $\neg x \varphi$, implying that $\varphi$ is false at the initial states.

- **Check** whether $\mathcal{C}$ is feasible.

- **Conclude** $\mathcal{D} \models \varphi$ iff $\mathcal{C}$ is infeasible.
Example

Consider the following system:

\[ D : 0, p \xrightarrow{} 1, \overline{p} \xrightarrow{} 2, p \]

For which we wish to verify the property \( \lozenge \square p \).
Example: Continued

Composing the system with the temporal tester $T(\lozenge \square p)$, we obtain:

with the justice requirements $\neg p \lor x$ and $x \lor \neg x$.

Eliminating all unreachable states and states with no successors, we are left with:

State 2 is eliminated because it does not have a path leading to a $\neg p \lor x$-state. Then state 1 is eliminated, having no successors. Finally, 0 is eliminated because it cannot reach a $\neg p \lor x$-state. Nothing is left, hence the system satisfies the property $\lozenge \square p$. 
Correctness of the Algorithms

Claim 9.
For an FDS $\mathcal{D}$ and temporal formula $\varphi$, $\mathcal{D} \models \varphi$ iff $C : \mathcal{D} \parallel T(\varphi) \parallel [\Theta : \neg x\varphi]$ is infeasible.

Proof:
The proof is based on the observation that every computation of the combined system $C$ is a computation of $\mathcal{D}$ which satisfies the negation of $\varphi$. Therefore, the existence of such a computation shows that not all computations of $\mathcal{D}$ satisfy $\varphi$, and therefore, $\varphi$ is not valid over $\mathcal{D}$. $\blacksquare$