Observations and Equivalence of Systems

Let $U \subseteq V$ be a subset of state variables and $s$ be a $V$-state. We denote by $s \downarrow_U$ the $U$-state, called the projection of $s$ on $U$, which is obtained by restricting the interpretation of variables to the variables in $U$.

For a $V$-state sequence

$$
\sigma : s_0, s_1, \ldots,
$$

we denote by $\sigma \downarrow_U$ the projected $U$-state sequence

$$
\sigma \downarrow_U : s_0 \downarrow_U, s_1 \downarrow_U, \ldots
$$

An $O$-state sequence $\Omega$ is called an observation of the FDS $D$ if $\Omega = \sigma \downarrow_O$ for some $\sigma$, a computation of $\sigma$. We denote by $\text{Obs}(D)$ the set of observations of FDS $D$.

Systems $D_1$ and $D_2$ are said to be equivalent, denoted $D_1 \sim D_2$, if their sets of observations are identical. That is,

$$
\text{Obs}(D_1) = \text{Obs}(D_2)
$$
Feasibility and Viability of Systems

An FDS $D$ is said to be feasible if $D$ has at least one computation.

A finite or infinite sequence of states is defined to be a run of an FDS $D$ if it satisfies the requirements of initiality and consecution but not necessarily any of the fairness requirements.

The FDS $D$ is defined to be viable if any finite run of $D$ can be extended to a computation of $D$.

Claim 7. Every FDS derived from an SPL program is viable.

Note that if $D$ is a viable system, such that its initial condition $\Theta_D$ is satisfiable, then $D$ is feasible.
Operations on FDS’s: Asynchronous Parallel Composition

Systems $\mathcal{D}_1$ and $\mathcal{D}_2$ are compatible if $V_1 \cap V_2 = \mathcal{O}_1 \cap \mathcal{O}_2$.

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

\[
\begin{align*}
V & = V_1 \cup V_2 \\
\mathcal{O} & = \mathcal{O}_1 \cup \mathcal{O}_2 \\
\Theta & = \Theta_1 \land \Theta_2 \\
\rho & = \left\{ \begin{array}{ll}
(\rho_1 \land \text{pres}(V_2 - V_1)) & \\
\lor & \\
(\rho_2 \land \text{pres}(V_1 - V_2)) & 
\end{array} \right. \\
\mathcal{J} & = \mathcal{J}_1 \cup \mathcal{J}_2 \\
\mathcal{C} & = \mathcal{C}_1 \cup \mathcal{C}_2
\end{align*}
\]

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 8. $\mathcal{D}(P_1 \parallel P_2) \sim \mathcal{D}(P_1) \parallel \mathcal{D}(P_2)$
Synchronous Parallel Composition

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

\[
V = V_1 \cup V_2 \\
\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2 \\
\Theta = \Theta_1 \land \Theta_2 \\
\rho = \rho_1 \land \rho_2 \\
\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_2 \\
\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2
\]

Synchronous parallel composition is useful for the modeling and verification of hardware designs. It is also useful for augmenting systems with auxiliary monitors.

Claim 9. Let $\sigma$ be an infinite $(V_1 \cup V_2)$-state sequence. Sequence $\sigma$ is a computation of $\mathcal{D}_1 \parallel \mathcal{D}_2$ iff $(\sigma \downarrow_{V_1}$ is a computation of $\mathcal{D}_1$ and $\sigma \downarrow_{V_2}$ is a computation of $\mathcal{D}_2$).
**Requirement Specification Language: Temporal Logic**

Assume an underlying (first-order) assertion language. The predicate $\text{at}_j \ell_i$, abbreviates the formula $\pi_j = \ell_i$, where $\ell_i$ is a location within process $P_j$.

A temporal formula is constructed out of state formulas (assertions) to which we apply the boolean operators $\neg$ and $\lor$ and various temporal operators, such as:

- $\square$ – always
- $\Diamond$ – eventually

A model for a temporal formula $p$ is an infinite sequence of states $\sigma : s_0, s_1, \ldots$, where each state $s_j$ provides an interpretation for the variables of $p$. 
Semantics of LTL

Given a model $\sigma$, we define the notion of a temporal formula $p$ holding at a position $j \geq 0$ in $\sigma$, denoted by $(\sigma, j) \models p$:

- For an assertion $p$,
  \[(\sigma, j) \models p \iff s_j \models p\]
  That is, we evaluate $p$ locally on state $s_j$.
- $(\sigma, j) \models \neg p \iff (\sigma, j) \not\models p$
- $(\sigma, j) \models p \lor q \iff (\sigma, j) \models p \text{ or } (\sigma, j) \models q$
- $(\sigma, j) \models \square p \iff (\sigma, k) \models p \text{ for all } k \geq j$
- $(\sigma, j) \models \Diamond p \iff (\sigma, k) \models p \text{ for some } k \geq j$

If $(\sigma, 0) \models p$ we say that $p$ holds over $\sigma$ and write $\sigma \models p$. $p$ is satisfiable if it holds over some model. $p$ is (temporally) valid if it holds over all models.

Formulas $p$ and $q$ are equivalent, denoted $p \sim q$, if $p \leftrightarrow q$ is valid. They are called congruent, denoted $p \cong q$, if $\square(p \leftrightarrow q)$ is valid. If $p \cong q$ then $p$ can be replaced by $q$ in any context.

We write $p \Rightarrow q$ as an abbreviation for $\square(p \rightarrow q)$. 
Reading Exercises

Following are some temporal formulas $\varphi$ and what they say about a sequence $\sigma : s_0, s_1, \ldots$ such that $\sigma \models \varphi$:

- $\Box p$ — All states within $\sigma$ satisfy $p$. Previously, we denoted this property by $\text{Inv}(p)$.

- $p \rightarrow \Diamond q$ — If $p$ holds at $s_0$, then $q$ holds at $s_j$ for some $j \geq 0$.

- $\Box (p \rightarrow \Diamond q)$ — Every $p$ is followed by a $q$. Also written as $p \Rightarrow \Diamond q$. Previously, we denoted this property by $p \leadsto q$.

- $\Box \Diamond q$ — The sequence $\sigma$ contains infinitely many $q$’s.

- $\Diamond \Box q$ — All but finitely many states in $\sigma$ satisfy $q$. Property $q$ eventually stabilizes.
Temporal Specification of Properties

Formula \( \varphi \) is \( D \)-valid, denoted \( D \models \varphi \), if all computations of \( D \) satisfy \( \varphi \). Such a formula specifies a property of \( D \).

Following is a temporal specification of the main properties of program MUX-SEM.

- **Mutual Exclusion** – No computation of the program can include a state in which process \( P_1 \) is at \( \ell_3 \) while \( P_2 \) is at \( m_3 \). Specifiable by the formula

\[ \square \neg (\text{at}_\ell \land \text{at}_m) \]

- **Accessibility** for \( P_1 \) – Whenever process \( P_1 \) is at \( \ell_2 \), it shall eventually reach it’s critical section at \( \ell_3 \). Specifiable by the formula

\[ \square (\text{at}_\ell \rightarrow \square \text{at}_\ell) \]
Full Temporal Logic – The Basic Operators

[] – Next  [ ] – Previous
[] – Until  [ ] – Since

Their semantics:

- \((\sigma, j) \models \Box p \iff (\sigma, j + 1) \models p\)
- \((\sigma, j) \models p \text{ } \mathcal{U} \text{ } q \iff \text{ for some } k \geq j, (\sigma, k) \models q, \text{ and for every } i \text{ such that } j \leq i < k, (\sigma, i) \models p\)
- \((\sigma, j) \models \neg \Box p \iff j > 0 \text{ and } (\sigma, j - 1) \models p\)
- \((\sigma, j) \models p \text{ } \mathcal{S} \text{ } q \iff \text{ for some } k \leq j, (\sigma, k) \models q, \text{ and for every } i \text{ such that } j \geq i > k, (\sigma, i) \models p\)

All other temporal operators can be defined in terms of these 4 as follows:

\[\Diamond p = 1 \mathcal{U} p\] – Eventually
\[\Box p = \neg \Diamond \neg p\] – Henceforth
\[p \mathcal{W} q = \Box p \lor (p \mathcal{U} q)\] – Waiting-for, Unless, Weak Until

\[\neg \Diamond p = 1 \mathcal{S} p\] – Sometimes in the past
\[\Box p = \neg \neg \Diamond \neg p\] – Always in the past
\[p \mathcal{B} q = \Box p \lor (p \mathcal{S} q)\] – Back-to, Weak Since
\[\neg \Box p = \neg \neg \Box \neg p\] – Weak Previous
Expressive Completeness

Every (propositional) temporal formula $\varphi$ can be translated into a first-order logic with monadic predicates over the naturals ordered by $<$ (1st-order theory of linear order).

For example, the 1st-order translation of $p \Rightarrow \diamondsuit q$ is

$$\forall t_1 \geq 0 : (p(t_1) \rightarrow \exists t_2 \geq t_1 : (q(t_2)))$$

Can every 1st-order formula be translated into temporal logic?

W. Kamp [Kamp68] has shown that the answer is negative if we only allow $\square$ and $\diamondsuit$ in our temporal formulas. But then proceeded to show that:

Claim 10. Every 1st-order formula can be translated into a temporal formula in the logic $\mathcal{L}(\mathcal{U} >, \mathcal{S} >)$.

[GPSS81] has shown that

Claim 11. Every 1st-order formula can be translated into a temporal formula in the logic $\mathcal{L}(\bigcirc, \mathcal{U})$.

This also shows that the past operators add no expressive power.
Classification of Formulas/Properties

A formula of the form $\Box p$ for some past formula $p$ is called a safety formula.

A formula of the form $\Box \lozenge p$ for some past formula $p$ is called a response formula.

An equivalent characterization is the form $p \Rightarrow \lozenge q$. The equivalence is justified by

$\Box (p \rightarrow \lozenge q) \sim \Box \lozenge ((\neg p) B q)$

Both formulas state that either there are infinitely many $q$’s, or there are no $p$’s, or there is a last $q$-position, beyond which there are no further $p$’s.

A property is classified as a safety/response property if it can be specified by a safety/response formula.

Every temporal formula is equivalent to a conjunction of a reactivity formulas, i.e.

$\bigwedge_{i=1}^{k} (\Box \lozenge p_i \lor \lozenge \Box q_i)$