Temporal Specification of Properties

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

**Temporal Specification of Properties**

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

\[
(\text{at } P_1 \leq m_3) \quad \text{\(\square\)}
\]

**Accessibility** for \( P_1 \) – Whenever process \( P_1 \) is at \( m_2 \), it shall eventually reach

\[
(\text{at } m_2 \rightarrow \text{at } m_3) \quad \text{\(\square\)}
\]

**Analysis of Reactive Systems**, NYU, Fall 2009

---

Following is a temporal specification of the main properties of program

\[
\varphi_0 \quad \text{naturally initially} \quad \varphi_1
\]

MUX-SM.
Also assume that we can Algorithmically check whether a given assertion is valid, i.e.,

\[(I)\varphi \lor (0)\varphi = (q)\varphi : q\varphi \quad (I)\varphi \land (0)\varphi = (q)\varphi : q\exists \varphi\]

boolean variable in.

Enumerative (explicit state) approach, by which we construct a graph containing all the reachable states of the system, and then apply graph theoretic algorithms to its analysis.

Symbolic approach, by which we continuously work with assertions which characterize sets of states.

Equivalently to I.

Theorem algorithms to its analysis.

There are two approaches to this process:

1. Enumerative (explicit state) approach, by which we construct a graph containing all the reachable states of the system, and then apply graph theoretic algorithms to its analysis.
2. Symbolic approach, by which we continuously work with assertions which characterize sets of states.

This is a process by which we Algorithmically check that a given finite-state FDS satisfies its temporal specification \(\varphi\).
\( ... \land R \diamond (R \diamond (R \diamond \varphi)) \land R \diamond (R \diamond \varphi) \land R \diamond \varphi \land \varphi = R \diamond \varphi \)

Successor Transformer:

The immediate successor transformer can be iterated to yield the eventual

\[
(I = x) \sim (I = x') \sim (I + x = x' \land 0 = x : x \in E) \sim = (I + x = x') \diamond (0 = x)
\]

For example

\[
\{ s \mid s \text{ is an } R\text{-successor of a } \varphi\text{-state} \} = \| R \diamond \varphi \|
\]

Obviously

\[
((\forall \lambda) R \land (\exists \lambda) \varphi : \lambda \in E) \sim = R \diamond \varphi
\]

Successor and Their Transitive Closure

For an assertion assertion and a bi-assertion \( (\forall \lambda) \varphi \) and \( (\exists \lambda) \varphi \) we define the existential
\[
\ldots \land ((\phi \land H) \land H) \land H \land ((\phi \land H) \land H) \land \phi \land H \land \phi = \phi \land H
\]

Prededessor transformer:
The immediate predecessor transformer can be iterated to yield the eventual:

\[
0 = x \land 1 = x \lor 1 + x = x : x \in = (1 = x) \land (1 + x = x)
\]

Example:

\[
\{ s \mid s \text{ is an } R\text{-predecessor of a } \phi\text{-state} \} = \| \phi \land H \|
\]

Obviously:

\[
(\neg \Lambda) \phi \lor (\neg \Lambda') H : \Lambda \in = \phi \land H
\]

Prededessor predicate transformer:
For an assertion \( (\Lambda A) \phi \) and a bi-assertion \( (\Lambda A) R \), we define the existential:

\[
\text{Predecessors and Their Transitive Closure}
\]

A. Pnueli
A set function is called monotonic if \( f \subseteq (1 + f) \) implies \( f \subseteq 1 \).

Restricting every set function \( f \subseteq X \) to be a solution.

Not every fix-point equation has a unique solution. Some, such as \( X = X \), have no solutions at all, while others, such as \( X - U = X \), have many solutions. In fact, \( X - U = X \) have many solutions, while others, such as \( X - U = X \), have no solutions.

In a similar way, we can define fix-points inclusions of the form \( (X)f \subseteq X \) and \( (X)f \subseteq X \).

A fix-point equation is an equation of the form

\[
\{ A \subseteq 1 + f | f \} = 1 + A
\]

which is a set function and is an unknown variable ranging over subsets of \( A \) if \( f \) is a set function mapping a set \( A \) of naturals into the set of their successors. Similarly, we can define \( \mathbb{N} = U \) \( f \) if we can define the set function \( U \). We consider set functions \( f \) with each element of the set \( U \) a set function mapping a set \( A \). For example,

\[
\begin{align*}
\text{(X-points)}
\end{align*}
\]
The Knaster-Tarski Theorem

A set $\mathcal{L}$ is said to be a minimal solution of the fix-point equation if $\mathcal{L} = (\mathcal{L})_f$ for a monotonic set function $f$. For a monotonically increasing function, the fix-point equation

$$X f = X$$

has a unique minimal solution $\mathcal{L}$. Furthermore, for the case that $\mathcal{L}$ is finite, $\mathcal{L}$ can be obtained as the union of the chain

$$(\mathcal{L})_f \bigcap \cdots \bigcap \mathcal{L}_0 = \mathcal{L}$$
Maximal Fix-points

\[\{0 \leq u \mid u \mathcal{L}\} = ((X \cdot \mathcal{L}) \cup X \cap \{1\}) \cdot X^a\]

For example, defining \( \mathcal{L} \), we have

\[\{X \ni x \mid x \mathcal{L}\} = X \cdot X^a\]

where

\[V - \emptyset = V\]

\[\forall (X)f \cdot X^a\]

An alternative approach to maximal fix-points defines

\[\bigcup_{i=0}^{\infty} f^i = (X)f \cdot X^a\]

the intersection always has a unique maximal solution which, for a finite \( V \) is given by \( (X)f = X \)

\( f \) variant of Knaster-Tarski will state that, for a monotonic function, the equation \( f \) can define maximal fix-points. The appropriate

Maximal Fix-points
Claim 4. [Invariance] Property \( p \) is valid over FDS over valid \( p \) \( \Box \) is unsatisfiable.

\[
\neg \Box \Diamond \Theta \quad \forall \Diamond \Theta \quad \forall \Diamond \Theta
\]

The algorithm returns an assertion characterizing all the initial states from which there exists a finite path leading to violation of \( p \). It returns the empty \((0)\) assertion if \( p \) is valid over \( D \).

As an example, we can use the following algorithm:

\[
(d \leftarrow (\Diamond \Theta \quad \Diamond \Theta)) \quad \forall \Diamond \Theta \quad \forall \Diamond \Theta
\]

**Algorithm INV** \((D; p)\):

1. Fix \( \text{preds} \) begin
2. if \( D \cap \text{preds} = 0 \) then \( \text{preds} := : p \_ (D \cap \text{preds}) \)
3. end
4. return \( D \cap \text{preds} \)

Assertion \( \forall \text{preds} \cup \Box \Theta \quad \forall \Theta \quad \forall \Theta \)

**Analysis of Reactive Systems**, NYU, Fall, 2009
There are several equivalent ways to compute the set of all eventual predecessors of an assertion $\Diamond$.

Equivalent Iterations:

$$\cdots \land ((\Diamond \Diamond d) \Diamond d) \land (\Diamond \Diamond d) \land \Diamond d \land \Diamond \Diamond d \land \Diamond d$$

$$\sim [X \Diamond d \land X =: X] (X)^{\text{Fix}} \quad \Diamond =: X$$

$$\sim [X \Diamond d \land \Diamond =: X] (X)^{\text{Fix}} \quad 0 =: X$$

$$\sim (X \Diamond d \land \Diamond).X^n$$

$$\sim \Diamond \Diamond d$$
Example: a Simpler MUX-SEM

Below, we present a simpler version of program MUX-SEM.

The semaphore instructions request \( y \) and release \( y \) respectively stand for

\[
\begin{align*}
\text{request } y & \quad \text{when } 0 =: y \quad \text{do } 1 =: y \\
\text{release } y & \quad \text{and } 1 =: y \\
\end{align*}
\]

A Puell

Analysis of Reactive Systems, NYU, Fall, 2009
We iterate as follows:

1. Let $\mathcal{C} = \overline{\nu} \lor \mathcal{C} = \overline{\nu}$ (false). We conclude that the initial condition $\mathcal{C}$ is not satisfied.

If we intersect $\mathcal{C}$ with the initial condition $\mathcal{O}$ we obtain:

$$d \sim (b \lor d) \land d$$

The last equivalence is due to the general property $p \land \neg p = \bot$.

$$\overline{\nu} \sim (0 = h \lor \mathcal{C} = \overline{\nu} \lor \mathcal{C} = \overline{\nu}) \land \overline{\nu} : \overline{\nu}$$

We iterate:

$$\mathcal{C} = \overline{\nu} = \overline{\nu}$$

$$0 = h \lor \mathcal{C} = \overline{\nu} \lor \mathcal{I} = h \lor \mathcal{L} = \overline{\nu} \lor \mathcal{C} = \overline{\nu}$$

$$\mathcal{C} = \overline{\nu} \lor \mathcal{C} = \overline{\nu} : \overline{\nu}$$
Symbolic Exploration Progresses in Layers

A. Pnueli

Analysis of Reactive Systems, NYU, Fall, 2009
We iterate as follows:

Illustrate Forward Exploration on MUX-SEM
Since last iteration does not intersect $C_1 \cap C_2$, we conclude

$\neg (C_1 \cap C_2)$.

Iteration 4 (Convergent):

Iteration 3:

Forward Exploration Continued