Copies of presentations and Lecture Notes will be available at

http://www.cs.nyu.edu/courses/spring04/G22.3033-017/index.htm

Thursdays, 5-7 PM

Amir Pnueli
Lecture 1: Preliminaries

A. Pnueli

The course will deal with topics concerning temporal verification of reactive systems which have not been considered in other courses.

Course Contents

1. The unified logic $\text{CTL}^\star$ in model checking and deductive verification.

2. Methods for parameterized verification, including regular model checking, network invariants, counter abstraction, and invisible auxiliary constructs.

3. Abstraction-aided verification — soundness and completeness of the method.

4. Predicate abstraction and abstraction refinement.

5. Methods for program synthesis from temporal specifications.

6. Syntheses of timed systems.

Possible additions to lecture 1:

- One of the available verification tools for model checking or deductive verification of reactive, real-time, or hybrid systems.

As part of the course work, students will be required to import, install, and demonstrate one of the available verification tools for model checking or deductive verification of reactive, real-time, or hybrid systems.

Advanced Topics in Reactive Verification, NYU, Spring, 2004
Lecture 1: Preliminaries

A. Pnueli

Fair Discrete Systems

\[ D = \{ V; O; J; C \} \]

\[ C = f h_p_1, q_1, \ldots, h_p_n, q_n ig \]

\[ J = f J_1, \ldots, J_k g \]

\[ V \] \{ A finite set of typed state variables. A \state s is an interpretation of \states. \]

\[ O \] \{ A set of observable variables. \]

\[ \Theta \] \{ An initial condition. A satisfying assertion that characterizes the initial states. \]

\[ J \] \{ A set of \states. \]

\[ D \] \{ A set of \states and \states. \]

\[ F \] \{ A set of fairness requirements. \]

\[ \delta \] \{ A transition relation. \]

\[ \cdot \cdot \] \{ A set of comparison (strong fairness) requirements. \]

\[ \cdot \cdot \] \{ An assertion \( (\Delta, \Delta)' \), referring to both unprimed (current) and primed (next) versions of the state variables. For example, \( x_0 + x = x \) corresponds to the assignment \( x : = x_0 + 1 \). \]

\[ \cdot \cdot \] \{ An initial condition. \]

\[ \cdot \cdot \] \{ A set of observable variables. \]

\[ \cdot \cdot \] \{ An initial set of typed state variables. \]

\[ \cdot \cdot \] \{ An initial condition. A satisfying assertion that characterizes the initial states. \]

\[ \cdot \cdot \] \{ A set of \states. \]

\[ \cdot \cdot \] \{ A set of \states. \]

\[ \cdot \cdot \] \{ An assertion \( \langle J, F, \delta, \Theta, O, \Delta \rangle = \Delta \). \]

\[ \cdot \cdot \] \{ Fair Discrete Systems \]
Let $D$ be an FDS for which the above components have been identified. The state $s^i$ is defined to be a $D$-successor of state $s$ if the state $s^i+1$ is a $D$-successor of the state $s^i$. Also contain infinitely many $b$-positions.

Compassion: For each $s \in C$, $s$ contains infinitely many $d$-positions.

Justice: For each $s \in S_1$, $s$ contains infinitely many $J$-positions.

Consecution: For each $s \in S$, $s$ contains infinitely many $J$-positions.

Initiality: $s_0$ is initial, i.e., $s_0 \models \Theta$.

We define a computation of $D$ to be an infinite sequence of states $\langle s_0, s_1, s_2, \ldots \rangle$. Let $\Phi \vdash \langle s_0, s_1, s_2, \ldots \rangle$.

Compositions
Lecture 1: Preliminaries

A. Pnueli

FS Operations: Asynchronous Parallel Composition

\( D_1 \parallel D_2 \sim D_1 \parallel D_2 \)

Claim 1. Asynchronous parallel composition represents the interleaving-based concurrency which is assumed in shared-variables models.

The predicate \( \text{pres} \) stands for the assertion \( \bigwedge \), implying that all the variables are preserved by the transition.

\[
\begin{align*}
\mathcal{C} \cap \mathcal{C}^1 &= \mathcal{C} \\
\mathcal{L} \cap \mathcal{L}^1 &= \mathcal{L} \\
((\exists \mathcal{A} - \mathcal{A}^1) \text{pres} \lor \exists \mathcal{D}) \land ((\exists \mathcal{A} - \mathcal{A}^1) \text{pres} \lor \exists \mathcal{D}^1) &= \mathcal{D} \\
\exists \Theta \lor \exists \Theta^1 &= \Theta \\
\exists \Theta \cap \exists \Theta^1 &= \Theta \\
\exists \mathcal{A} \cap \exists \mathcal{A}^1 &= \mathcal{A}
\end{align*}
\]

where \( \bigwedge \), is given by \( D_1 \parallel D_2 \), is denoted by \( D_1 \parallel D_2 \).

By \( D_1 \parallel D_2 \), is denoted the asynchronous parallel composition of the compatible systems \( D_1 \) and \( D_2 \), denoted \( D_1 \parallel D_2 \).

The systems \( D_1 \) and \( D_2 \) are compatible if \( \mathcal{O}^1 \cup \mathcal{O}^2 = \mathcal{O}^1 \cup \mathcal{O}^2 \).

Advanced Topics in Reactive Verification, NYU, Spring, 2004
The synchronous parallel composition of the compatible systems $D_1$ and $D_2$, denoted $D_1 \parallel D_2$, is given by the FDS $\mathcal{F} = \langle V; O; J; C \rangle$ where

- $V = V_1 \cup V_2$,
- $O = O_1 \cup O_2$,
- $J = J_1 \cup J_2$,
- $C = C_1 \cup C_2$.

Synchronous parallel composition is used for the construction of an observer $O$ which observes and evaluates the behavior of an observed system $D$. Running $D_1 \parallel O$, we let $D_1$ behave as usual, while $O$ observes its behavior.

Note that the synchronous parallel composition of two non-blocking FDS's can yield a blocking FDS.

**Synchronous Parallel Composition**
A.Pnueli

Requirement Specification Language: the Temporal Logic

Assume an underlying (first-order) assertion language \( \mathcal{L} \). The predicate at-\( \xi \), abbreviates the formula \( \exists \xi \), where \( \xi \) is a location within process \( P \).

A temporal formula is constructed out of assertions to which we apply the

\[ \begin{align*}
& \text{Path quantifiers: } \mathcal{E}, \mathcal{A}, \mathcal{E}^f, \text{ and } \mathcal{A}^f.
& \text{Temporal operators: } \\
& \text{ - Previous } S \quad \text{ - Since } b \quad \text{ - Until } W
& \text{ - Next } N \quad \text{ - Waiting-for, Unless } \mathcal{W}
& \text{ - Back-to, } \mathcal{B}
& \text{ Boolean operators: } \land, \lor, \text{ and } \neg.
\end{align*} \]
Additional temporal operators can be defined in terms of the basic ones as follows:

- Always in the past: \( 0 \boxdot d = d \square \)
- Sometimes in the past: \( d \lozenge I = d \Diamond \)
- Henceforth: \( 0 \lozenge d = d \Box \)
- Eventually: \( d \lozenge I = d \Diamond \)

**Derived Temporal Operators**
There are two types of sub-formulas in $\text{CTL}^*$:

- **State formulas (interpreted over states):**
  - Every assertion in $L$ is a state formula.
  - If $p$ is a path formula, then $E_p$, $A_p$, $E_f p$, and $A_f p$ are state formulas.
  - If $p$ and $q$ are state formulas then so are $\neg p$, $p \lor q$, and $p \land q$.

Examples: $p$ and $A \square (p \rightarrow \Diamond q)$ are state formulas.
A state formula is a $\text{CTL^*}$ formula. Path formulas which contain no path quantifiers are sometimes referred to as $\text{LTL}$ formulas.

Examples: $p$ and $\mathsf{E}r$ are path formulas.

Any state formula is a $\text{CTL^*}$ formula. Path formulas which contain no path quantifiers are sometimes referred to as $\text{LTL}$ formulas.

Path formulas (interpreted over state sequences): $\text{CTL^*}$ syntax (2/2)
In Pictures
The sequence $o$ being maximal means that either $o$ is infinite, or $o$ is finite, in which case $o$ satisfies the requirements of justice and compassion.

The sequence $o$ being maximal means that either $o$ is infinite, or $o$ is finite, in which case $o$ satisfies the requirements of justice and compassion.

Recall that a computation of a $D$ is an infinite run which satisfies the requirements of justice and compassion.

A state $s$ is said to be reachable if it participates in some run of $D$. A state $s$ is feasible if it participates in some computation of $D$. A $D$ is called deadlock-free if every reachable state has a $D$-successor. Note that all runs of a deadlock-free $D$ are infinite. It can be shown that all $D$'s derived from programs are deadlock-free.

Let $D$ be an $D$. A run of $D$ is a maximal sequence of states $s_0, s_1, \ldots, s_n$ of $D$.

**Runs, Reachable, and Feasible States**

**A. Pnueli**

Advanced Topics in Reactive Verification, NYU, Spring 2004
The semantics of $\mathsf{CTL}^*$ formulas is defined inductively as follows:

Let $\mathcal{D}$ be a run of $\mathcal{F}$ and $\varsigma_0, \varsigma_1, \ldots$ be the states in $\mathcal{F}$. Then, for $\psi$ a formula, we write $[\psi]_{\varsigma}$ to denote the $\psi$-value of $\varsigma$, the $\psi$-value of $\varsigma$.

The term path as synonymous to a run of an $\mathcal{F}$, let $\Psi : \varsigma_0, \varsigma_1, \ldots$ be a run of $\mathcal{F}$. We interpret $\mathsf{CTL}^*$ formulas over (the computation structure of) an $\mathcal{F}$ in the language of $\mathsf{CTL}^*$.
Interpretation of State Formulas

State formulas are interpreted over states in $D$. We define the notion of a state $s$ as a state holding at a state in $D$. We denote the state formulas are interpreted over states in $D$. State formulas are interpreted similarly to $\forall$ and $\exists$ respectively:

$$s = [s'] \forall \text{satisfying } \emptyset \in \emptyset$$

For state formulas and positions $p \in \text{runs } \in D$ and position $s = [s'] \forall \text{satisfying } \emptyset \in \emptyset$.

For a path formula $\forall$ and $\exists$ are defined similarly to $\forall$ and $\exists$ respectively, replacing path (run) by computation.

For an assertion $d$ of $\exists$ and $\forall$, as follows:

$$d \models s \quad \text{as follows:}$$

Interpretation of State Formulas
Path formulas are interpreted over runs of $D$. We define the notion of a path formula $\phi$, denoted $|\phi|$, at position $j \in \text{trans}(D)$ of a run $\rho$ of $D$. We define the notion of a path formula $\phi$, denoted $\rho |\phi|$, at position $j \in \text{trans}(D)$ of a run $\rho$ of $D$.

For state formulas $\phi$, $\rho |\phi| = (\rho |\phi|)\rho |\phi|$

For path formulas $\phi$ and $\psi$, $\rho |\phi| \land \rho |\psi| = (\rho |\phi|)\rho |\phi|$

For a state formula $\phi$, $\rho |\phi| = (\rho |\phi|)\rho |\phi|$

For path formulas $\phi$ and $\psi$, $\rho |\phi| \land \rho |\psi| = (\rho |\phi|)\rho |\phi|$

Path formulas are interpreted over runs of $D$. We define the notion of a path formula $\phi$, denoted $|\phi|$, at position $j \in \text{trans}(D)$ of a run $\rho$ of $D$. We define the notion of a path formula $\phi$, denoted $\rho |\phi|$, at position $j \in \text{trans}(D)$ of a run $\rho$ of $D$.

Interpretation of Path Formulas (1/2)
Interpretation of Path Formulas

Let $\phi$ be a CTL* formula. We say that $\phi$ holds on some model $\mathcal{M}$ if it holds on all reachable states, denoted $\phi = \mathcal{M}$.

A CTL* formula is called satisfiable if it holds on some model. A CTL* formula is called valid if it holds on all models.

Let $\phi$ be a CTL* formula. We say that $\phi$ holds on a model $\mathcal{M}$ if its holds on all reachable states.

For all $n \geq 0$, for all $d = (s, f, \nu, \alpha)$:

$\phi \supseteq \nu \geq 0$:

$\phi \supseteq \nu = n$

$\phi \supseteq \nu > n$

$\phi \supseteq \nu \neq n$

$\phi \supseteq \nu = \bot$

For path formulas and $\phi$:

$\phi \not\supseteq \nu \geq 0$:

$\phi \not\supseteq \nu = \top$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu = \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$

$\phi \not\supseteq \nu \neq \bot$

$\phi \not\supseteq \nu > \bot$

$\phi \not\supseteq \nu < \top$

$\phi \not\supseteq \nu < \bot$
Consider the following system $D$:

\[
\begin{align*}
D_0 &: p \\
D_1 &: 2 \\
D_2 &: 0
\end{align*}
\]

\[
D_j = E (p^E p) \quad \exists \phi \neq a
\]

- There exists a reachable state which has one predecessor satisfying $p$ and another predecessor satisfying $p$. Indeed, $2$ is such a state.

\[
D_0 = E (p^E p) \quad \exists \phi \neq a
\]

- There exists a reachable state which has one predecessor satisfying $p$ and another predecessor satisfying $p$. Indeed, $2$ is such a state.

\[
D_0 = E (p^E p) \quad \exists \phi \neq a
\]

- There exists an unreach state on the same path.

\[
D_0 = E (p^E p) \quad \exists \phi \neq a
\]

Examples
Following every $p$, $b$ precedes $a$: $b \prec a \cdot$

Strongly $b$ precedes $a$: $b \prec a \cdot$

$b$ is not guaranteed, but $a$ cannot happen without a preceding $b$. $b \prec a \cdot$

Note that $b$ precedes $a$. $b \prec a \cdot$

Every $b$ is preceded by a cause: $d \prec b \cdot$

$\Box$ stabilizes: $b \diamond \Box \cdot$

$\forall$ but finitely many states at $\diamond$ satisfy $b$. Property $b$ eventually satisfies: $b \Box \cdot$

The sequence $\diamond$ contains infinitely many $b$'s: $b \diamond \Box \cdot$

Every $b$ is followed by a $d$. Can also be written as $b \diamond \Box \cdot$

If holds at $s_0$, then holds at $s$ for some $s > s_0$ holds at $s$: $b \diamond \Box \cdot$

Following are some temporal formulas and what do they say about an infinite path:

- $\diamond | (s_0, s_1, \ldots) \cdot$
- $\forall \Diamond$ such that $\diamond$:

**Reading Exercises**
Temporal Specification of Properties

A valid formula $\varphi$ specifies a property of $D$.

Following is a temporal specification of the main properties of program $\text{mux-sem}$.

1. **Mutual Exclusion**
   - No computation of the program includes a state in which $P_1$ is at $c_3$ while $P_2$ is at $m_3$. Specifiable by $\varphi = \Box (\text{at-}c_3 \lor \text{at-}m_3)$.

2. **Accessibility**
   - Whenever process $P_1$ is at $c_2$, it shall eventually reach its critical section at $c_3$. Specifiable by $\varphi = \Diamond (\text{at-}c_3) \Rightarrow \Box (\text{at-}c_2 \lor \text{at-}c_3)$.

3. **Advanced Topics in Reactive Verification**
   - $P_1$ - $P_2$ whenever $P_1$ is at $c_3$, it shall eventually reach its critical section at $c_3$. Specifiable by $\varphi = \Diamond (\text{at-}c_3) \Rightarrow \Box (\text{at-}c_2 \lor \text{at-}c_3)$.

4. **Accessibility**
   - Whenever process $P_1$ is at $c_2$, it shall eventually reach its critical section at $c_3$. Specifiable by $\varphi = \Diamond (\text{at-}c_3) \Rightarrow \Box (\text{at-}c_2 \lor \text{at-}c_3)$.

5. **Mutual Exclusion**
   - No computation of the program includes a state in which $P_1$ is at $c_3$ while $P_2$ is at $m_3$. Specifiable by $\varphi = \Box (\text{at-}c_3 \lor \text{at-}m_3)$.

6. **Temporal Specification of Properties**

   $\varphi$ is a $\text{CTL}^*$ formula that specifies a property of $D$. 
Fragmentsof

CTL

{Pathquantifiersandtemporaloperatorsalwaysappearinthecombination

Q\text{T},whereQisapathquantifierandTisatemporaloperator.

A^\mathcal{F},whereAisa pathquantifierandTisa temporaloperator.

\mathcal{A},where\mathcal{A}isapathquantifierandTisa temporaloperator.

\mathcal{Q},where\mathcal{Q}isa pathquantifierandTisa temporaloperator.

\mathcal{G},where\mathcal{G}isa pathquantifierandTisa temporaloperator.

\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.

\mathcal{F}^\mathcal{F},where\mathcal{F}isa pathquantifierandTisa temporaloperator.}
Comparison Between the Fragments

Properties in $\mathcal{LTL} - \mathcal{CTL}$:

- $\Box \Diamond \Box d$. Continuations is not equivalent to $\forall d \Box d$. On every path, there exists a point beyond which $d$ holds on all continuations.

Properties in $\mathcal{CTL} - \mathcal{LTL}$:

- $\forall (b \Box \land d \Box) \Box$. On every path, either continuously $d$ or continuously $b$.
- $(b \Diamond \Box \leftarrow d \Diamond \Box) \Box$. Compassion.

Properties in $\mathcal{LTL} - \mathcal{CTL}$:

- $\forall$ term. It is always possible for the program to terminate.
- $d \Diamond \Box$. It is possible to reach a $d$-state.

A. Pnueli

Advanced Topics in Reactive Verification, NYU, Spring, 2004

Lecture 1: $\mathcal{CTL}$
Consider the following PDS $\mathcal{A}$:

$$\mathcal{A} \nvdash p$$

This system satisfies $\mathcal{A}$ but $\mathcal{A} \nvdash p$. However, the computation 0 has no position which satisfies $\mathcal{A} p$. Both types satisfy $\mathcal{D}$.

System $\mathcal{D}$ has two types of computations:

- $\mathcal{D}_1$:
  - $p$,
- $\mathcal{D}_2$:
  - $\Box p$.