CS 181U Applied Logic

Lecture 9

Computation Tree Logic
νSMV peculiarities

The process keyword will be deprecated. Ignore the warning.
νSMV peculiarities

The process keyword will be deprecated. Ignore the warning.

Comments indicated with two dashes: --
νSMV peculiarities

The process keyword will be deprecated. Ignore the warning.

Comments indicated with two dashes: -

MODULE main, and processes P0 and P1 are three separate threads. P0 and P1 are subthreads of main.

```
MODULE proc(id, ...)
  ...
MODULE main
  ...
p0 = proc(0, ...)
p1 = proc(1, ...)
  ...
```

Use FAIRNESS running in proc specification.
νSMV peculiarities

The process keyword will be deprecated. Ignore the warning.

Comments indicated with two dashes: –

MODULE main, and processes P0 and P1 are three separate threads. P0 and P1 are subthreads of main.

MODULE proc(id, ...) ... 

MODULE main ... 

p0 = proc(0, ...) 

p1 = proc(1, ...) 

Use FAIRNESS running in proc specification.
Interesting Quote

If what is exactly stated can be done by a machine, the residue of the uniquely human becomes coextensive with the linguistic qualities that interfere with precise specification—ambiguity, metaphoric play, multiple encoding, and allusive exchanges between one symbol system and another. The uniqueness of human behavior thus becomes assimilated to the ineffability of language, and the common ground that humans and machines share is identified with the univocality of an instrumental language that has banished ambiguity from its lexicon.

-N. Katherine Hayles

How we Became Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics
Reminder

Linear Temporal Logic (LTL)
We will assign symbols for expressing temporal system requirements like always \((G)\), eventually \((F)\), next \((X)\), until \((U)\), and a few more. We will give a formal and unambiguous semantics to these symbols.

Transition Systems
We will learn a formal system of specifying transition systems (which we often depict as a transition diagram).

Concurrency Concepts
Safety, liveness, mutual exclusion, …

Verification Software
Symbolic Model Verifier (NuSMV)
Reminder

Linear Temporal Logic (LTL)
We will assign symbols for expressing temporal system requirements like \textit{always} ($G$), \textit{eventually} ($F$), \textit{next} ($X$), \textit{until} ($U$), and a few more. We will give a formal and unambiguous semantics to these symbols.

Transition Systems
We will learn a formal system of specifying transition systems (which we often depict as a transition diagram).

Concurrency Concepts
Safety, liveness, mutual exclusion, ...

Verification Software
Symbolic Model Verifier (NuSMV)

We did all this.
Next

**Computation Tree Logic (CTL)**

We will learn a different way to write temporal properties of systems.

**Verification Software + CTL**

Symbolic Model Verifier (NuSMV) with CTL
Computation Tree Logic (CTL)
We will learn a different way to write temporal properties of systems.

Verification Software + CTL
Symbolic Model Verifier (NuSMV) with CTL
Remember the big picture

- Reactive System Code
  - satisfies
  - $\models$
  - Requirements

- Transition System
  - satisfies
  - $\models$
  - Temporal Logic Formula $\phi$

Model Checking
Linear vs Branching Time Logic
Linear vs Branching Time Logic

Some paths of $\mathcal{M}$

0 \xrightarrow{p,q} 1 \xrightarrow{q,r} 0 \xrightarrow{p,q} 1 \xrightarrow{q,r} 0 \xrightarrow{p,q} \ldots
Linear vs Branching Time Logic

Some paths of $\mathcal{M}$

\[(\circlearrowright 0, p, q) \xrightarrow{p, q} 1 \xrightarrow{q, r} 0 \xrightarrow{p, q} 1 \xrightarrow{q, r} 0 \xrightarrow{p, q} \cdots \]

\[(\circlearrowright 0, p, q) \xrightarrow{p, q} 1 \xrightarrow{q, r} 0 \xrightarrow{p, q} 2 \xrightarrow{r} 2 \xrightarrow{r} \cdots \]
Linear vs Branching Time Logic

Some paths of $M$

1. $0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow \cdots$
   - $p, q \rightarrow q, r \rightarrow p, q \rightarrow q, r \rightarrow p, q$

2. $0 \rightarrow 1 \rightarrow 0 \rightarrow 2 \rightarrow 2 \rightarrow \cdots$
   - $p, q \rightarrow q, r \rightarrow p, q \rightarrow r \rightarrow r$

3. $0 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow \cdots$
   - $p, q \rightarrow q, r \rightarrow r \rightarrow r \rightarrow r$
Linear vs Branching Time Logic

Some paths of $\mathcal{M}$

- $0 \overset{p,q}{\rightarrow} 1 \overset{q,r}{\rightarrow} 0 \overset{q,r}{\rightarrow} 1 \overset{p,q}{\rightarrow} 0 \overset{p,q}{\rightarrow} \cdots$
- $0 \overset{p,q}{\rightarrow} 1 \overset{q,r}{\rightarrow} 0 \overset{p,q}{\rightarrow} 2 \overset{r}{\rightarrow} 2 \overset{r}{\rightarrow} \cdots$
- $0 \overset{p,q}{\rightarrow} 1 \overset{q,r}{\rightarrow} 2 \overset{r}{\rightarrow} 2 \overset{r}{\rightarrow} 2 \overset{r}{\rightarrow} \cdots$
- $0 \overset{p,q}{\rightarrow} 2 \overset{r}{\rightarrow} 2 \overset{r}{\rightarrow} 2 \overset{r}{\rightarrow} 2 \overset{r}{\rightarrow} \cdots$
Linear vs Branching Time Logic

LTL Model Checking

Some paths of $\mathcal{M}$

- $0 \xrightarrow{p, q} 1 \xrightarrow{q, r} 0 \xrightarrow{p, q} 1 \xrightarrow{q, r} 0 \xrightarrow{p, q} \ldots$
- $0 \xrightarrow{p, q} 1 \xrightarrow{q, r} 0 \xrightarrow{p, q} 2 \xrightarrow{r} 2 \xrightarrow{r} \ldots$
- $0 \xrightarrow{p, q} 1 \xrightarrow{q, r} 2 \xrightarrow{r} 2 \xrightarrow{r} 2 \xrightarrow{r} \ldots$
- $0 \xrightarrow{p, q} 2 \xrightarrow{r} 2 \xrightarrow{r} 2 \xrightarrow{r} 2 \xrightarrow{r} \ldots$
Some paths of $\mathcal{M}$

\[
\begin{align*}
    &0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow \ldots \ldots \\
    &0 \rightarrow 1 \rightarrow 0 \rightarrow 2 \rightarrow 2 \rightarrow \ldots \ldots \\
    &0 \rightarrow 1 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow \ldots \ldots \\
    &0 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow 2 \rightarrow \ldots \ldots
\end{align*}
\]

$LTL$ formula

$\mathcal{M} \models \phi \iff \forall \pi \ [\pi \models \phi]$
Linear vs Branching Time Logic

\[0\] \quad p, q

\[1\] \quad q, r

\[2\] \quad r
Linear vs Branching Time Logic

Computation tree for $M$

$p, q$ $igcirc$

$p, q$
Linear vs Branching Time Logic

Computation tree for $\mathcal{M}$
Linear vs Branching Time Logic

Computation tree for $M$
Linear vs Branching Time Logic

Computation tree for $\mathcal{M}$

- States: 0, 1, 2
- Transitions: $p, q, r, q, r$
Linear vs Branching Time Logic

Computation tree for $M$
Linear vs Branching Time Logic

Computing tree for $M$

Computation Tree Logic (CTL) expresses properties of "alternative timelines".
Computation Tree Logic (CTL) expresses properties of “alternative timelines”.

$$\mathcal{M} \models \phi \iff \text{CTL formula}$$
Suppose $\alpha$ and $\beta$ are LTL formulas. Suppose $p_i$ is a propositional atom. Then the following are all LTL formulas.

$$
\top \quad \bot \quad p_i \\
\neg \alpha \quad \alpha \lor \beta \quad \alpha \land \beta \quad \alpha \rightarrow \beta
$$
Computation Tree Logic Syntax

Suppose $\alpha$ and $\beta$ are LTL formulas.
Suppose $p_i$ is a propositional atom.
Then the following are all LTL formulas.

\[
\begin{align*}
\top & \quad \bot & \quad p_i \\
\neg \alpha & \quad \alpha \lor \beta & \quad \alpha \land \beta & \quad \alpha \rightarrow \beta
\end{align*}
\]

We use the same temporal operators: $G, F, X, U$
We attach path quantifiers, $A$ (inevitably) or $E$ (possibly), to each temporal operator.

\[
\begin{align*}
AG & \quad AF & \quad AX & \quad AU \\
EG & \quad EF & \quad EX & \quad EU
\end{align*}
\]
Computation Tree Logic Semantics
Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$. 
Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

We say that a state, $s$, of $M$ satisfies a CTL formula $\phi$, written $s \models \phi$, according to the following recursive informal definition:
Computation Tree Logic Semantics

Recall the state labelling function, \( L(s) \).

We say that a state, \( s \), of \( M \) satisfies a CTL formula \( \phi \), written \( s \models \phi \), according to the following recursive informal definition:

- Base case: If \( \phi \) is atomic then \( s \models \phi \) iff \( \phi \in L(s) \).
Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

We say that a state, $s$, of $M$ satisfies a CTL formula $\phi$, written $s \models \phi$, according to the following recursive informal definition:

- **Base case:** If $\phi$ is atomic then $s \models \phi$ iff $\phi \in L(s)$.
- If $\phi = \alpha \land \beta$, then $s \models \phi$ iff $s \models \alpha$ and $s \models \beta$.
  Similar rules apply if $\phi$ is one of $\neg$, $\lor$, $\rightarrow$. 
Computation Tree Logic Semantics

Recall the state labelling function, \( L(s) \).

We say that a state, \( s \), of \( \mathcal{M} \) satisfies a CTL formula \( \phi \), written \( s \models \phi \), according to the following recursive informal definition:

- **Base case:** If \( \phi \) is atomic then \( s \models \phi \) iff \( \phi \in L(s) \).
- If \( \phi = \alpha \land \beta \), then \( s \models \phi \) iff \( s \models \alpha \) and \( s \models \beta \). Similar rules apply if \( \phi \) is one of \( \neg, \lor, \rightarrow \).
- If \( \phi \) is an \( A \)-operator, then \( s \models \phi \) iff all paths starting at \( s \) satisfy the ‘LTL formula’ made by removing the \( A \) symbol.
Computation Tree Logic Semantics

Recall the state labelling function, $L(s)$.

We say that a state, $s$, of $M$ satisfies a CTL formula $\phi$, written $s \models \phi$, according to the following recursive informal definition:

- Base case: If $\phi$ is atomic then $s \models \phi$ iff $\phi \in L(s)$.

- If $\phi = \alpha \land \beta$, then $s \models \phi$ iff $s \models \alpha$ and $s \models \beta$. Similar rules apply if $\phi$ is one of $\neg, \lor, \to$.

- If $\phi$ is an $A$-operator, then $s \models \phi$ iff all paths starting at $s$ satisfy the ‘LTL formula’ made by removing the $A$ symbol.

- If $\phi$ is an $E$-operator, then $s \models \phi$ iff there exists a path starting at $s$ satisfy the ‘LTL formula’ made by removing the $E$ symbol.
We just defined what it means for a state to satisfy a CTL property, $s \models \phi$. 
We just defined what it means for a state to satisfy a CTL property, $s \models \phi$.

Now we want to define what it means for a transition system $\mathcal{M}$ to satisfy a CTL property.
Computation Tree Logic Semantics

We just defined what it means for a state to satisfy a CTL property, \( s \models \phi \).

Now we want to define what it means for a transition system \( \mathcal{M} \) to satisfy a CTL property.

\[ \mathcal{M} \models \phi \iff \forall s \in I \ s \models \phi \]

CTL Model Checking
The idea of $AG\phi$: inevitably always $\phi$
The idea of $AF\phi$: inevitably eventually $\phi$
Computation Tree Logic Semantics

The idea of $EG\phi$: possibly always $\phi$
Computation Tree Logic Semantics

The idea of $EF\phi$: possibly eventually $\phi$
Computation Tree Logic Semantics

The idea of $EX\phi$: possibly next $\phi$
Computation Tree Logic Semantics

The idea of $AX\phi$: inevitably next $\phi$
Computation Tree Logic Semantics

The idea of $\phi \mathcal{A} \mathcal{U} \psi$: inevitably $\phi$ until $\psi$
The idea of $\phi \text{AU} \psi$: inevitably $\phi$ until $\psi$

Fill in your answer in class.
Computation Tree Logic Semantics

The idea of $\phi EU \psi$: possibly $\phi$ until $\psi$
The idea of $\phi EU \psi$: possibly $\phi$ until $\psi$

Fill in your answer in class.
Computation Tree Logic Example Properties

Computation tree for $M$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$

1. $p, q$
2. $q, r$
Computation Tree Logic Example Properties

$\mathcal{M} \models p \land q$?
Computation Tree Logic Example Properties

Computation tree for $\mathcal{M}$

$\mathcal{M} \models p \land q$?

$\mathcal{M} \models \neg r$?
Computation Tree Logic Example Properties

Computation tree for $\mathcal{M}$

$\mathcal{M} \models p \land q$?

$\mathcal{M} \models \neg r$?

$\mathcal{M} \models EX(q \land r)$?
Computation Tree Logic Example Properties

\[ \mathcal{M} \models p \land q ? \]
\[ \mathcal{M} \models \neg r ? \]
\[ \mathcal{M} \models EX(q \land r) ? \]
\[ \mathcal{M} \models AX(q \land r) ? \]

Computation tree for \( \mathcal{M} \)
Computation Tree Logic Example Properties

\[ \mathcal{M} \models p \land q ? \]
\[ \mathcal{M} \models \neg r ? \]
\[ \mathcal{M} \models EX(q \land r) ? \]
\[ \mathcal{M} \models AX(q \land r) ? \]
\[ \mathcal{M} \models \neg AX(q \land r) ? \]
Computation Tree Logic Example Properties

\[ \mathcal{M} \models p \land q \ ? \]
\[ \mathcal{M} \models \neg r \ ? \]
\[ \mathcal{M} \models EX(q \land r) \ ? \]
\[ \mathcal{M} \models AX(q \land r) \ ? \]
\[ \mathcal{M} \models \neg AX(q \land r) \ ? \]
\[ \mathcal{M} \models \neg EF(p \land r) \ ? \]
Computation Tree Logic Example Properties

\[ \mathcal{M} \models p \land q ? \]
\[ \mathcal{M} \models \neg r ? \]
\[ \mathcal{M} \models EX(q \land r) ? \]
\[ \mathcal{M} \models AX(q \land r) ? \]
\[ \mathcal{M} \models \neg AX(q \land r) ? \]
\[ \mathcal{M} \models \neg EF(p \land r) ? \]
\[ \mathcal{M} \models p A U r ? \]
\[ \mathcal{M} \models \neg (p \land q) E U r ? \]
Computation Tree Logic Example Properties

\[ \mathcal{M} \models p \land q \]?
\[ \mathcal{M} \models \neg r \]?
\[ \mathcal{M} \models \text{EX}(q \land r) \]?
\[ \mathcal{M} \models \text{AX}(q \land r) \]?
\[ \mathcal{M} \models \neg \text{AX}(q \land r) \]?
\[ \mathcal{M} \models \neg \text{EF}(p \land r) \]?
\[ \mathcal{M} \models \text{EG}\neg r? \]
\[ \mathcal{M} \models \text{AF}q？ \]
\[ \mathcal{M} \models p \text{ AU } r \]
\[ \mathcal{M} \models \neg(p \land q) \text{ EU } r \]?