Linear Temporal Logic (LTL)

- Grammar of *well formed formulae* (wff) $\phi$

$$
\phi ::= p \quad \text{(Atomic formula: } p \in AP) \\
| \neg \phi \quad \text{(Negation)} \\
| \phi_1 \lor \phi_2 \quad \text{(Disjunction)} \\
| X\phi \quad \text{(successor)} \\
| F\phi \quad \text{(sometimes)} \\
| G\phi \quad \text{(always)} \\
| [\phi_1 U \phi_2] \quad \text{(Until)}
$$

- Details differ from Prior’s tense logic – but similar ideas

- Semantics define when $\phi$ true in model $M$
  - where $M = (S, S_0, R, L)$ – a Kripke structure
  - notation: $M \models \phi$ means $\phi$ true in model $M$
  - model checking algorithms compute this (when decidable)
\( M \models \phi \) means “wff \( \phi \) is true in model \( M \)”

- If \( M = (S, S_0, R, L) \) then
  \[ \pi \text{ is an } M\text{-path starting from } s \text{ iff } \text{Path } R \ s \ \pi \]

- If \( M = (S, S_0, R, L) \) then we define \( M \models \phi \) to mean:
  \( \phi \) is true on all \( M\)-paths starting from a member of \( S_0 \)

- We will define \( \llbracket \phi \rrbracket_M(\pi) \) to mean
  \( \phi \) is true on the \( M\)-path \( \pi \)

- Thus \( M \models \phi \) will be formally defined by:
  \[ M \models \phi \iff \forall \pi \ s. s \in S_0 \land \text{Path } R \ s \ \pi \Rightarrow \llbracket \phi \rrbracket_M(\pi) \]

- It remains to actually define \( \llbracket \phi \rrbracket_M \) for all wffs \( \phi \)
Definition of $\llbracket \phi \rrbracket_M(\pi)$

- $\llbracket \phi \rrbracket_M(\pi)$ is the application of function $\llbracket \phi \rrbracket_M$ to path $\pi$
  - thus $\llbracket \phi \rrbracket_M : (\mathbb{N} \to S) \to \mathbb{B}$

- Let $M = (S, S_0, R, L)$
  - $\llbracket \phi \rrbracket_M$ is defined by structural induction on $\phi$

- $\llbracket p \rrbracket_M(\pi) = p \in L(\pi \ 0)$
- $\llbracket \neg \phi \rrbracket_M(\pi) = \neg(\llbracket \phi \rrbracket_M(\pi))$
- $\llbracket \phi_1 \lor \phi_2 \rrbracket_M(\pi) = \llbracket \phi_1 \rrbracket_M(\pi) \lor \llbracket \phi_2 \rrbracket_M(\pi)$
- $\llbracket X\phi \rrbracket_M(\pi) = \llbracket \phi \rrbracket_M(\pi \downarrow 1)$
- $\llbracket F\phi \rrbracket_M(\pi) = \exists i. \llbracket \phi \rrbracket_M(\pi \downarrow i)$
- $\llbracket G\phi \rrbracket_M(\pi) = \forall i. \llbracket \phi \rrbracket_M(\pi \downarrow i)$
- $\llbracket [\phi_1 U \phi_2] \rrbracket_M(\pi) = \exists i. \llbracket \phi_2 \rrbracket_M(\pi \downarrow i) \land \forall j. j < i \Rightarrow \llbracket \phi_1 \rrbracket_M(\pi \downarrow j)$

- We look at each of these semantic equations in turn
\[ \llbracket p \rrbracket_M(\pi) = p(\pi 0) \]

- Assume \( M = (S, S_0, R, L) \)

- We have: \( \llbracket p \rrbracket_M(\pi) = p \in L(\pi 0) \)
  - \( p \) is an atomic property, i.e. \( p \in AP \)
  - \( \pi : \mathbb{N} \to S \) so \( \pi 0 \in S \)
  - \( \pi 0 \) is the first state in path \( \pi \)
  - \( p \in L(\pi 0) \) is true iff atomic property \( p \) holds of state \( \pi 0 \)

- \( \llbracket p \rrbracket_M(\pi) \) means \( p \) holds of the first state in path \( \pi \)

- \( T, F \in AP \) with \( T \in L(s) \) and \( F \notin L(s) \) for all \( s \in S \)
  - \( \llbracket T \rrbracket_M(\pi) \) is always true
  - \( \llbracket F \rrbracket_M(\pi) \) is always false
\[ \neg \phi \models_M (\pi) = \neg (\models_M (\phi \models_M (\pi))) \]

\[ (\phi_1 \lor \phi_2) \models_M (\pi) = (\models_M (\phi_1 \models_M (\pi)) \lor (\models_M (\phi_2 \models_M (\pi))) \]

- \( \neg \phi \models_M (\pi) \) is true iff \( \models_M (\phi \models_M (\pi)) \) is not true

- \( (\phi_1 \lor \phi_2) \models_M (\pi) \) is true iff \( \models_M (\phi_1 \models_M (\pi)) \) is true or \( \models_M (\phi_2 \models_M (\pi)) \) is true
\([\mathbf{X}\phi]_M(\pi) = [\phi]_M(\pi\downarrow 1)\]

- \([\mathbf{X}\phi]_M(\pi) = [\phi]_M(\pi\downarrow 1)\)
  - \(\pi\downarrow 1\) is \(\pi\) with the first state chopped off
    - \(\pi\downarrow 1(0) = \pi(1 + 0) = \pi(1)\)
    - \(\pi\downarrow 1(1) = \pi(1 + 1) = \pi(2)\)
    - \(\pi\downarrow 1(2) = \pi(1 + 2) = \pi(3)\)
    - \(\vdots\)

- \([\mathbf{X}\phi]_M(\pi)\) true iff \([\phi]_M\) true starting at the second state of \(\pi\)
\[ [F \phi]_M(\pi) = \exists i. [\phi]_M(\pi \downarrow i) \]

- \( [F \phi]_M(\pi) = \exists i. [\phi]_M(\pi \downarrow i) \)
  - \( \pi \downarrow i \) is \( \pi \) with the first \( i \) states chopped off
    - \( \pi \downarrow i(0) = \pi(i + 0) = \pi(i) \)
    - \( \pi \downarrow i(1) = \pi(i + 1) \)
    - \( \pi \downarrow i(2) = \pi(i + 2) \)
    - \( \vdots \)
  - \( [\phi]_M(\pi \downarrow i) \) true iff \( [\phi]_M \) true starting \( i \) states along \( \pi \)

- \( [F \phi]_M(\pi) \) true iff \( [\phi]_M \) true starting somewhere along \( \pi \)
- “\( F \phi \)” is read as “sometimes \( \phi \)”
\[ \mathbf{[G\phi]}_M(\pi) = \forall i. \mathbf{[\phi]}_M(\pi \downarrow i) \]

- \( \mathbf{[G\phi]}_M(\pi) = \forall i. \mathbf{[\phi]}_M(\pi \downarrow i) \)
  - \( \pi \downarrow i \) is \( \pi \) with the first \( i \) states chopped off
  - \( \mathbf{[\phi]}_M(\pi \downarrow i) \) true iff \( \mathbf{[\phi]}_M \) true starting \( i \) states along \( \pi \)

- \( \mathbf{[G\phi]}_M(\pi) \) true iff \( \mathbf{[\phi]}_M \) true starting anywhere along \( \pi \)

- "\( \mathbf{G\phi} \)" is read as "always \( \phi \)" or "globally \( \phi \)"

- \( M \models \mathbf{AG} \rho \) defined earlier: \( M \models \mathbf{AG} \rho \iff M \models \mathbf{G} (\rho) \)

- \( \mathbf{G} \) is definable in terms of \( \mathbf{F} \) and \( \neg: \mathbf{G\phi} = \neg (\mathbf{F} (\neg \phi)) \)
  \[ \mathbf{[\neg (\mathbf{F} (\neg \phi))]}_M(\pi) = \neg (\mathbf{[\neg (\mathbf{F} (\neg \phi))]}_M(\pi)) \]
  \[ = \neg (\exists i. \mathbf{[\neg \phi]}_M(\pi \downarrow i)) \]
  \[ = \neg (\exists i. \neg (\mathbf{[\phi]}_M(\pi \downarrow i))) \]
  \[ = \forall i. \mathbf{[\phi]}_M(\pi \downarrow i) \]
  \[ = \mathbf{[G\phi]}_M(\pi) \]
\[ [[\phi_1 U \phi_2]]_M(\pi) = \exists i. [[\phi_2]]_M(\pi \downarrow i) \land \forall j. j < i \Rightarrow [[\phi_1]]_M(\pi \downarrow j) \]

- \[ [[\phi_1 U \phi_2]]_M(\pi) = \exists i. [[\phi_2]]_M(\pi \downarrow i) \land \forall j. j < i \Rightarrow [[\phi_1]]_M(\pi \downarrow j) \]
  - \[ [[\phi_2]]_M(\pi \downarrow i) \text{ true iff } [[\phi_2]]_M \text{ true starting } i \text{ states along } \pi \]
  - \[ [[\phi_1]]_M(\pi \downarrow j) \text{ true iff } [[\phi_1]]_M \text{ true starting } j \text{ states along } \pi \]

- \[ [[\phi_1 U \phi_2]]_M(\pi) \text{ is true iff } [[\phi_2]]_M \text{ is true somewhere along } \pi \text{ and up to then } [[\phi_1]]_M \text{ is true} \]

- "[[\phi_1 U \phi_2]]" is read as "\phi_1 \text{ until } \phi_2"

- \[ F \text{ is definable in terms of } [\neg U \neg]: F_{\phi} = [[T U \phi]] \]
  \[
  [[T U \phi]]_M(\pi) \\
  = \exists i. [[\phi]]_M(\pi \downarrow i) \land \forall j. j < i \Rightarrow [[T]]_M(\pi \downarrow j) \\
  = \exists i. [[\phi]]_M(\pi \downarrow i) \land \forall j. j < i \Rightarrow true \\
  = \exists i. [[\phi]]_M(\pi \downarrow i) \land true \\
  = \exists i. [[\phi]]_M(\pi \downarrow i) \\
  = [[F_{\phi}]]_M(\pi) \]
Review of Linear Temporal Logic (LTL)

 Grammar of well formed formulae (wff) $\phi$

$$\begin{align*}
\phi & ::= p \quad \text{(Atomic formula: } p \in AP) \\
& \mid \neg \phi \quad \text{(Negation)} \\
& \mid \phi_1 \lor \phi_2 \quad \text{(Disjunction)} \\
& \mid X\phi \quad \text{(successor)} \\
& \mid F\phi \quad \text{(sometimes)} \\
& \mid G\phi \quad \text{(always)} \\
& \mid [\phi_1 U \phi_2] \quad \text{(Until)}
\end{align*}$$

$M \models \phi$ means $\phi$ holds on all $M$-paths

- $M = (S, S_0, R, L)$
- $[\phi]_M(\pi)$ means $\phi$ is true on the $M$-path $\pi$
- $M \models \phi \iff \forall \pi \ s. \ s \in S_0 \land \text{Path } R s \pi \Rightarrow [\phi]_M(\pi)$
LTL examples

- “DeviceEnabled holds infinitely often along every path”
  \[ G(F \text{DeviceEnabled}) \]

- “Eventually the state becomes permanently Done“
  \[ F(G \text{Done}) \]

- “Every Req is followed by an Ack”
  \[ G(\text{Req} \Rightarrow F \text{Ack}) \]
  Number of Req and Ack may differ - no counting

- “If Enabled infinitely often then Running infinitely often”
  \[ G(F \text{Enabled}) \Rightarrow G(F \text{Running}) \]

- “An upward going lift at the second floor keeps going up if a passenger requests the fifth floor”
  \[ G(\text{AtFloor2} \land \text{DirectionUp} \land \text{RequestFloor5} \Rightarrow [\text{DirectionUp} U \text{AtFloor5}]) \]
A property not expressible in LTL

Let \( AP = \{ P \} \) and consider models \( M \) and \( M' \) below

\[
M = (\{ s_0, s_1 \}, \{ s_0 \}, \{( s_0, s_0 ), ( s_0, s_1 ), ( s_1, s_1 ) \}, L)
\]

\[
M' = (\{ s_0 \}, \{ s_0 \}, \{( s_0, s_0 ) \}, L)
\]

where: \( L = \lambda s. \text{if } s = s_0 \text{ then } \{ \} \text{ else } \{ P \} \)

- Every \( M' \)-path is also an \( M \)-path
- So if \( \phi \) true on every \( M \)-path then \( \phi \) true on every \( M' \)-path
- Hence in LTL for any \( \phi \) if \( M \models \phi \) then \( M' \models \phi \)
- Consider \( \phi_P \Leftrightarrow \text{“can always reach a state satisfying } P \text{”} \)
  - \( \phi_P \) holds in \( M \) but not in \( M' \)
  - but in LTL can’t have \( M \models \phi_P \) and not \( M' \models \phi_P \)
  - hence \( \phi_P \) not expressible in LTL
LTL expressibility

“can always reach a state satisfying $P$”

- In LTL $M \models \phi$ says $\phi$ holds of all paths of $M$
- LTL formulae $\phi$ are evaluated on paths . . . . path formulae
- Want to say that from any state there exists a path to some state satisfying $p$
  - $\forall s. \exists \pi. \text{Path } R s \pi \land \exists i. p \in L(\pi(i))$
  - but this isn’t expressible in LTL (see slide 57)
- CTL properties are evaluated at a state . . . state formulae
  - they can talk about both some or all paths
  - starting from the state they are evaluated at
Computation Tree Logic (CTL)

- LTL formulae $\phi$ are evaluated on paths . . . . path formulae
- CTL formulae $\psi$ are evaluated on states . . state formulae

Syntax of CTL well-formed formulae:

$$\psi ::= p \quad \text{(Atomic formula } p \in AP)$$
$$\quad \quad \quad \neg \psi \quad \text{(Negation)}$$
$$\quad \quad \quad \psi_1 \land \psi_2 \quad \text{(Conjunction)}$$
$$\quad \quad \quad \psi_1 \lor \psi_2 \quad \text{(Disjunction)}$$
$$\quad \quad \quad \psi_1 \Rightarrow \psi_2 \quad \text{(Implication)}$$
$$\quad \quad \quad AX \psi \quad \text{(All successors)}$$
$$\quad \quad \quad EX \psi \quad \text{(Some successors)}$$
$$\quad \quad \quad A[\psi_1 U \psi_2] \quad \text{(Until – along all paths)}$$
$$\quad \quad \quad E[\psi_1 U \psi_2] \quad \text{(Until – along some path)}$$
Semantics of CTL

Assume \( M = (S, S_0, R, L) \) and then define:

\[
\begin{align*}
[p]_M(s) &= p \in L(s) \\
[\neg \psi]_M(s) &= \neg ([\psi]_M(s)) \\
[\psi_1 \land \psi_2]_M(s) &= [\psi_1]_M(s) \land [\psi_2]_M(s) \\
[\psi_1 \lor \psi_2]_M(s) &= [\psi_1]_M(s) \lor [\psi_2]_M(s) \\
[\psi_1 \Rightarrow \psi_2]_M(s) &= [\psi_1]_M(s) \Rightarrow [\psi_2]_M(s) \\
[A \psi]_M(s) &= \forall s'. R s s' \Rightarrow [\psi]_M(s') \\
[E \psi]_M(s) &= \exists s'. R s s' \land [\psi]_M(s') \\
[A[\psi_1 U \psi_2]]_M(s) &= \forall \pi. \text{Path } R s \pi \\
&\quad \Rightarrow \exists i. [\psi_2]_M(\pi(i)) \\
&\quad \land \forall j. j < i \Rightarrow [\psi_1]_M(\pi(j)) \\
[E[\psi_1 U \psi_2]]_M(s) &= \exists \pi. \text{Path } R s \pi \\
&\quad \land \exists i. [\psi_2]_M(\pi(i)) \\
&\quad \land \forall j. j < i \Rightarrow [\psi_1]_M(\pi(j))
\end{align*}
\]
The defined operator \( \text{AF} \)

- Define \( \text{AF} \psi = \text{A}[\text{T U} \psi] \)

- \( \text{AF} \psi \) true at \( s \) iff \( \psi \) true somewhere on every \( R \)-path from \( s \)

\[
\begin{align*}
\llbracket \text{AF} \psi \rrbracket_M(s) &= \llbracket \text{A}[\text{T U} \psi] \rrbracket_M(s) \\
&= \forall \pi. \text{Path } R \ s \ \pi \\
&\quad \Rightarrow \\
&\quad \exists i. \llbracket \psi \rrbracket_M(\pi(i)) \land \forall j. j < i \Rightarrow \llbracket \text{T} \rrbracket_M(\pi(j)) \\
&= \forall \pi. \text{Path } R \ s \ \pi \\
&\quad \Rightarrow \\
&\quad \exists i. \llbracket \psi \rrbracket_M(\pi(i)) \land \forall j. j < i \Rightarrow \text{true} \\
&= \forall \pi. \text{Path } R \ s \ \pi \Rightarrow \exists i. \llbracket \psi \rrbracket_M(\pi(i))
\end{align*}
\]
The defined operator \( \textsf{EF} \)

- Define \( \textsf{EF} \psi = \text{E}[\textsf{U} \psi] \)

- \( \textsf{EF} \psi \) true at \( s \) iff \( \psi \) true somewhere on some \( R \)-path from \( s \)

\[
[\text{EF} \psi]_M(s) = \left[ \text{E}[\textsf{U} \psi] \right]_M(s)
\]

\[
= \exists \pi. \text{Path } R s \pi \\
\land \\
\exists i. \left[ \psi \right]_M(\pi(i)) \land \forall j. j < i \Rightarrow \left[ \textsf{T} \right]_M(\pi(j))
\]

\[
= \exists \pi. \text{Path } R s \pi \\
\land \\
\exists i. \left[ \psi \right]_M(\pi(i)) \land \forall j. j < i \Rightarrow \text{true}
\]

\[
= \exists \pi. \text{Path } R s \pi \land \exists i. \left[ \psi \right]_M(\pi(i))
\]

- “can reach a state satisfying \( \rho \)” is \( \textsf{EF} \rho \)
The defined operator $AG$

- Define $AG\psi = \neg EF(\neg \psi)$

- $AG\psi$ true at $s$ iff $\psi$ true everywhere on every $R$-path from $s$

$$\[AG\psi\]_M(s) = \[\neg EF(\neg \psi)\]_M(s)$$
$$= \neg (\[EF(\neg \psi)\]_M(s))$$
$$= \neg (\exists \pi. \text{Path } R s \pi \land \exists i. \[\neg \psi\]_M(\pi(i)))$$
$$= \forall \pi. \neg (\exists i. \[\neg \psi\]_M(\pi(i)))$$
$$= \forall \pi. \neg (\exists i. \neg \[\psi\]_M(\pi(i)))$$
$$= \forall \pi. \neg \neg \[\psi\]_M(\pi(i))$$
$$= \forall \pi. \neg \neg \[\psi\]_M(\pi(i))$$
$$= \forall \pi. \text{Path } R s \pi \Rightarrow \forall i. \[\psi\]_M(\pi(i))$$

- $AG\psi$ means $\psi$ true at all reachable states

- $[AG(p)]_M(s) \equiv \forall s'. R^* s s' \Rightarrow p \in L(s')$

- “can always reach a state satisfying $p$” is $AG(EF p)$
The defined operator $\text{EG}$

- Define $\text{EG} \psi = \neg \text{AF}(\neg \psi)$

- $\text{EG} \psi$ true at $s$ iff $\psi$ true everywhere on some $R$-path from $s$

$$\llbracket \text{EG} \psi \rrbracket_M(s) = \llbracket \neg \text{AF}(\neg \psi) \rrbracket_M(s)$$
$$= \neg \llbracket \text{AF}(\neg \psi) \rrbracket_M(s)$$
$$= \neg (\forall \pi. \text{Path } R s \pi \Rightarrow \exists i. \llbracket \neg \psi \rrbracket_M(\pi(i)))$$
$$= \neg (\forall \pi. \text{Path } R s \pi \Rightarrow \exists i. \neg \llbracket \psi \rrbracket_M(\pi(i)))$$
$$= \exists \pi. \neg (\text{Path } R s \pi \Rightarrow \exists i. \neg \llbracket \psi \rrbracket_M(\pi(i)))$$
$$= \exists \pi. \text{Path } R s \pi \land \neg (\exists i. \neg \llbracket \psi \rrbracket_M(\pi(i)))$$
$$= \exists \pi. \text{Path } R s \pi \land \forall i. \neg \llbracket \psi \rrbracket_M(\pi(i))$$
$$= \exists \pi. \text{Path } R s \pi \land \forall i. \llbracket \psi \rrbracket_M(\pi(i))$$
The defined operator \( A[\psi_1 \ W \ \psi_2] \)

- \( A[\psi_1 \ W \ \psi_2] \) is a ‘partial correctness’ version of \( A[\psi_1 \ U \ \psi_2] \)
- It is true at \( s \) if along all \( R \)-paths from \( s \):
  - \( \psi_1 \) always holds on the path, or
  - \( \psi_2 \) holds sometime on the path, and until it does \( \psi_1 \) holds

- Define

\[
\llbracket A[\psi_1 \ W \ \psi_2] \rrbracket_M(s) = \llbracket \neg E[(\psi_1 \land \neg \psi_2) \ U (\neg \psi_1 \land \neg \psi_2)] \rrbracket_M(s)
\]

\[
= \neg \llbracket E[(\psi_1 \land \neg \psi_2) \ U (\neg \psi_1 \land \neg \psi_2)] \rrbracket_M(s)
\]

\[
= \neg (\exists \pi. \text{Path } R \ s \ \pi
\]

\[
\land
\]

\[
\exists i. \llbracket \neg \psi_1 \land \neg \psi_2 \rrbracket_M(\pi(i))
\]

\[
\land
\]

\[
\forall j. j < i \Rightarrow \llbracket \psi_1 \land \neg \psi_2 \rrbracket_M(\pi(j))
\]

- Exercise: understand the next two slides!
A[ψ₁ W ψ₂] continued (1)

Continuing:

\[\neg(\exists \pi. \text{Path } R \ s \ \pi)\]
\[\land\]
\[\exists i. \lbrack \neg \psi₁ \land \neg \psi₂ \rbrack_M(\pi(i)) \land \forall j. j<i \Rightarrow \lbrack \psi₁ \land \neg \psi₂ \rbrack_M(\pi(j))\]

= \[\forall \pi. \neg(\text{Path } R \ s \ \pi)\]
\[\land\]
\[\exists i. \lbrack \neg \psi₁ \land \neg \psi₂ \rbrack_M(\pi(i)) \land \forall j. j<i \Rightarrow \lbrack \psi₁ \land \neg \psi₂ \rbrack_M(\pi(j))\]

= \[\forall \pi. \text{Path } R \ s \ \pi\]
\[\Rightarrow\]
\[\neg(\exists i. \lbrack \neg \psi₁ \land \neg \psi₂ \rbrack_M(\pi(i)) \land \forall j. j<i \Rightarrow \lbrack \psi₁ \land \neg \psi₂ \rbrack_M(\pi(j)))\]

= \[\forall \pi. \text{Path } R \ s \ \pi\]
\[\Rightarrow\]
\[\forall i. \neg\lbrack \neg \psi₁ \land \neg \psi₂ \rbrack_M(\pi(i)) \lor \neg(\forall j. j<i \Rightarrow \lbrack \psi₁ \land \neg \psi₂ \rbrack_M(\pi(j)))\]
\[ A[\psi_1 W \psi_2] \text{ continued (2)} \]

- Continuing:

\[ = \forall \pi. \text{Path } R s \pi \]
\[ \Rightarrow \]
\[ \forall i. \neg [\neg \psi_1 \land \neg \psi_2]_M(\pi(i)) \lor \neg (\forall j. j < i \Rightarrow [\psi_1 \land \neg \psi_2]_M(\pi(j))) \]
\[ = \forall \pi. \text{Path } R s \pi \]
\[ \Rightarrow \]
\[ \forall i. \neg (\forall j. j < i \Rightarrow [\psi_1 \land \neg \psi_2]_M(\pi(j))) \lor \neg [\neg \psi_1 \land \neg \psi_2]_M(\pi(i)) \]
\[ = \forall \pi. \text{Path } R s \pi \]
\[ \Rightarrow \]
\[ \forall i. (\forall j. j < i \Rightarrow [\psi_1 \land \neg \psi_2]_M(\pi(j))) \Rightarrow [\psi_1 \lor \psi_2]_M(\pi(i)) \]

- Exercise: explain why this is \([A[\psi_1 W \psi_2]]_M(s)\)?

  - this exercise illustrates the subtlety of writing CTL!
Sanity check: \( A[\psi \ W \ F] = AG \psi \)

- From last slide:
  \[
  [A[\psi_1 \ W \ \psi_2]]_M(s)
  = \forall \pi. \text{ Path } R s \pi
  \Rightarrow \forall i. (\forall j. j < i \Rightarrow [\psi_1 \land \neg \psi_2]_M(\pi(j))) \Rightarrow [\psi_1 \lor \psi_2]_M(\pi(i))
  \]

- Set \( \psi_1 \) to \( \psi \) and \( \psi_2 \) to \( F \):
  \[
  [A[\psi \ W \ F]]_M(s)
  = \forall \pi. \text{ Path } R s \pi
  \Rightarrow \forall i. (\forall j. j < i \Rightarrow [\psi \land \neg F]_M(\pi(j))) \Rightarrow [\psi \lor F]_M(\pi(i))
  \]

- Simplify:
  \[
  [A[\psi \ W \ F]]_M(s)
  = \forall \pi. \text{ Path } R s \pi \Rightarrow \forall i. (\forall j. j < i \Rightarrow [\psi]_M(\pi(j))) \Rightarrow [\psi]_M(\pi(i))
  \]

- By induction on \( i \):
  \[
  [A[\psi \ W \ F]]_M(s) = \forall \pi. \text{ Path } R s \pi \Rightarrow \forall i. [\psi]_M(\pi(i))
  \]

Exercises

1. Describe the property: \( A[T \ W \ \psi] \).
2. Describe the property: \( \neg E[\neg \psi_2 \ U \ (\neg \psi_1 \lor \psi_2)] \).
3. Define \( E[\psi_1 \ W \ \psi_2] = E[\psi_1 \ U \ \psi_2] \lor EG\psi_1 \).
   Describe the property: \( E[\psi_1 \ W \ \psi_2] \)?
Recall model behaviour computation tree

- Atomic properties are true or false of individual states
- General properties are true or false of whole behaviour
- Behaviour of $(S, R)$ starting from $s \in S$ as a tree:

  - A path is shown in red
  - Properties may look at all paths, or just a single path
    - CTL: Computation Tree Logic (all paths from a state)
    - LTL: Linear Temporal Logic (a single path)
Summary of CTL operators (primitive + defined)

- **CTL formulae:**
  
  - $\rho$ (Atomic formula - $\rho \in AP$)
  - $\neg \psi$ (Negation)
  - $\psi_1 \land \psi_2$ (Conjunction)
  - $\psi_1 \lor \psi_2$ (Disjunction)
  - $\psi_1 \implies \psi_2$ (Implication)
  - $AX\psi$ (All successors)
  - $EX\psi$ (Some successors)
  - $AF\psi$ (Somewhere – along all paths)
  - $EF\psi$ (Somewhere – along some path)
  - $AG\psi$ (Everywhere – along all paths)
  - $EG\psi$ (Everywhere – along some path)
  - $A[\psi_1 U \psi_2]$ (Until – along all paths)
  - $E[\psi_1 U \psi_2]$ (Until – along some path)
  - $A[\psi_1 W \psi_2]$ (Unless – along all paths)
  - $E[\psi_1 W \psi_2]$ (Unless – along some path)
Example CTL formulae

- **EF(\(Started \land \neg Ready\))**
  
  It is possible to get to a state where \(Started\) holds but \(Ready\) does not hold

- **AG(\(Req \Rightarrow AF Ack\))**

  If a request \(Req\) occurs, then it will eventually be acknowledged by \(Ack\)

- **AG(\(AF DeviceEnabled\))**

  \(DeviceEnabled\) is always true somewhere along every path starting anywhere: i.e. \(DeviceEnabled\) holds infinitely often along every path

- **AG(\(EF Restart\))**

  From any state it is possible to get to a state for which \(Restart\) holds

  Can’t be expressed in LTL!
More CTL examples (1)

- \( \text{AG}(\text{Req} \Rightarrow \text{A}[\text{Req U Ack}]) \)
  
  If a request \( \text{Req} \) occurs, then it continues to hold, until it is eventually acknowledged.

- \( \text{AG}(\text{Req} \Rightarrow \text{AX}(\text{A}[\neg\text{Req U Ack}])) \)
  
  Whenever \( \text{Req} \) is true either it must become false on the next cycle and remains false until \( \text{Ack} \), or \( \text{Ack} \) must become true on the next cycle.

  Exercise: is the \text{AX} necessary?

- \( \text{AG}(\text{Req} \Rightarrow (\neg\text{Ack} \Rightarrow \text{AX}(\text{A}[\text{Req U Ack}]))) \)
  
  Whenever \( \text{Req} \) is true and \( \text{Ack} \) is false then \( \text{Ack} \) will eventually become true and until it does \( \text{Req} \) will remain true.

  Exercise: is the \text{AX} necessary?
More CTL examples (2)

- \( \mathbf{AG} \left( \mathbf{Enabled} \Rightarrow \mathbf{AG} \left( \mathbf{Start} \Rightarrow \mathbf{A}[\neg\mathbf{Waiting} \mathbf{U} \mathbf{Ack}] \right) \right) \)
  
  If \( \mathbf{Enabled} \) is ever true then if \( \mathbf{Start} \) is true in any subsequent state then \( \mathbf{Ack} \) will eventually become true, and until it does \( \mathbf{Waiting} \) will be false

- \( \mathbf{AG} \left( \neg\mathbf{Req}_1 \land \neg\mathbf{Req}_2 \Rightarrow \mathbf{A}[\neg\mathbf{Req}_1 \land \neg\mathbf{Req}_2 \mathbf{U} (\mathbf{Start} \land \neg\mathbf{Req}_2)] \right) \)
  
  Whenever \( \mathbf{Req}_1 \) and \( \mathbf{Req}_2 \) are false, they remain false until \( \mathbf{Start} \) becomes true with \( \mathbf{Req}_2 \) still false

- \( \mathbf{AG} (\mathbf{Req} \Rightarrow \mathbf{AX}(\mathbf{Ack} \Rightarrow \mathbf{AF} \neg\mathbf{Req})) \)
  
  If \( \mathbf{Req} \) is true and \( \mathbf{Ack} \) becomes true one cycle later, then eventually \( \mathbf{Req} \) will become false
Some abbreviations

- \( AX_i \psi \equiv AX(AX(\cdots(AX \psi)\cdots)) \)
  - \( i \) instances of \( AX \)
  - \( \psi \) is true on all paths \( i \) units of time later

- \( ABF_{i..j} \psi \equiv AX_i(\psi \lor AX(\psi \lor \cdots AX(\psi \lor AX \psi)\cdots)) \)
  - \( j - i \) instances of \( AX \)
  - \( \psi \) is true on all paths sometime between \( i \) units of time later and \( j \) units of time later

- \( AG(Req \Rightarrow AX(Ack_1 \land ABF_{1..6}(Ack_2 \land A[Wait U Reply]))) \)
  - One cycle after \( Req \), \( Ack_1 \) should become true, and then \( Ack_2 \) becomes true 1 to 6 cycles later and then eventually \( Reply \) becomes true, but until it does \( Wait \) holds from the time of \( Ack_2 \)

- More abbreviations in ‘Industry Standard’ language PSL