Computation Tree Logic (CTL)
CTL

Syntax

• $P$ - a set of atomic propositions, every $p \in P$ is a CTL formula.
• $f, g$, CTL formulae, then so are $\neg f$, $f \land g$, $EXf$, $A[fUg]$, $E[fUg]$

$E, A$ – path quantifiers, $X, U, G, F$ – temporal operators

Interpreted over a tree of states:

• $EXf$ - $f$ holds in some immediate successor
• $A[fUg]$ - every path has a prefix that satisfies $[fUg]$
**CTL Model (Kripke model)**

\[ M = (S, s_I, R, L) \]

- \( S \) is a finite set of states, \( s_I \in S \) is the initial state
- \( R \subseteq S \times S \) s.t. \( \forall u \in S. \exists v \in S. (u,v) \in R \) (total)
- \( L : S \rightarrow 2^{\text{AP}} \)

Example: \( \text{AP} = \{p,q,r\} \)
Example: XR Control Program

\{\text{@open, stop}\}

\{\text{Tout, @open, stop}\}

\{\text{Tout, Tin}\}

\{\text{Tin, open↓, @close}\}

\{\text{Tin, open!, @close}\}

\{\text{Tout, Tin}\}

\{\text{close!, go}\}

\{\text{open!}\}

\{\text{Tin, @close}\}
The infinite computation tree spanned by a model $M=(S,s_i,R,L)$

- root $s_0=s_i$
- $s \rightarrow t$ is an arc in the tree iff $(s,t) \in R$.

A path is: $s_0, s_1, s_2, \ldots \in S^\omega$ s.t. $\forall i. (s_i, s_{i+1}) \in R$
CTL Semantics

w.r.t a given model \((S, s_I, R, L)\) and a state \(s \in S\):

\[
\begin{align*}
s \models p & \iff p \in L(s) \\
s \models \neg f & \iff \text{not } s \models f \\
s \models f \land g & \iff s \models f \text{ and } s \models g \\
s \models \text{EX} f & \iff \exists s'. (s, s') \in R \text{ and } s' \models f \\
s \models \text{A}[f \cup g] & \iff \text{for every path } (s_0, s_1, \ldots) \text{ s.t } s_0 = s, \\
& \quad \exists k \geq 0. \ s_k \models g \land \forall i. 0 \leq i < k \implies s_i \models f \\
s \models \text{E}[f \cup g] & \iff \text{for some path } (s_0, s_1, \ldots) \text{ s.t } s_0 = s, \\
& \quad \exists k \geq 0. \ s_k \models g \land \forall i. 0 \leq i < k \implies s_i \models f
\end{align*}
\]

- \(f\) is \textit{satisfiable} iff there exists a model \((S, s_I, R, L)\) such that \(s_I \models f\)
- \(f\) is \textit{valid} iff \(f\) is \textit{satisfiable} by every model \((S, s_I, R, L)\)
Derived Operators

AX(f) ≡ ¬EX¬f  
       f holds at all next states

AF(f) ≡ A[true U f]  
       f holds in the future of every path

EF(f) ≡ E[true U f]  
       f holds in the future of some path

AG(f) ≡ ¬EF(¬f)  
       every state in every path satisfies f

EG(f) ≡ ¬AF(¬f)  
       for some path every state satisfies f
A note on temporal wff

Syntax: \( \text{EXf, A}[f\text{Ug}], \text{E}[f\text{Ug}] \)

- \( \text{E, A} \) – path quantifiers
- \( \text{X, U, G, F} \) – temporal operators

Structure of a formula: \((\text{path})(\text{temporal})[\text{formula}]\)

- \( \text{E, A} \) – path quantifiers
- \( \text{X, U, G, F} \) – temporal operators

♥ \( \text{EF(Gu)} \) is not a wff, only \( \text{EF(EGu)} \) or \( \text{EF(AGu)} \)

♥ \( \text{A(fUg)} \) cannot be derived from \( \text{E(fUg)} \) since \( \neg \text{E}(\neg(f\text{Ug})) \) is not wff (since \( \neg(f\text{Ug}) \) is not wff)
Properties Expressed in CTL

- Every *req* is followed by *ack* (not 1-1): $\text{AG}(\text{req} \rightarrow \text{AF ack})$
- It is possible to get to a state where *started* holds but *ready* does not hold.
  \[
  \text{EF}(\text{started} \land \neg \text{ready})
  \]
- From any state it is possible to get to the *restart* state.
  \[
  \text{AG EF restart}
  \]
- Processes P,Q are not in their critical section simultaneously
  \[
  \text{AG}(\neg(\text{Pc} \land \text{Qc}))
  \]
- A process that asks to enter the critical section, eventually gets there
  \[
  \text{AG}(\text{PE} \rightarrow \text{AF(Pc)})
  \]
- Processes strictly alternate in access to the critical section
  \[
  \text{AG(Pc} \rightarrow \text{A}(\text{PcU}(\neg \text{Pc} \land \text{A}(\neg \text{PcUQc}))))
  \]
Assertions:

• 50 seconds minimal delay between trains.
  \[ \text{AG}(\text{Tin} \Rightarrow \text{AX} \neg \text{Tin} \land \text{AXAX} \neg \text{Tin} \land \ldots \land \text{AX..AX} \neg \text{Tin}) \]

• It takes a train 6 seconds to arrive at the signal
  \[ \text{AG}(\text{Tin} \Rightarrow \text{AX}(\text{AX}(\text{AX}(\text{AX}(\text{AX}(\text{AX}(\text{AtSignal}))))))) \]
  however, we could write \[ \text{AG}(\text{Tin} \Rightarrow \text{AX}^6(\text{AtSignal}) \]
  like we did in LTL
LTL vs. CTL

- Different models - A CTL tree defines a subset of models of LTL
- Expressivity – Define LTL satisfied on Kripke model if satisfied by every path.

- LTL cannot distinguish behaviors that generate same path, therefore cannot express ‘possibility’.
- CTL is sensitive to the branching structure

Thus, “it is always possible to get tea after inserting a coin” is expressible in CTL: \( \text{AG(coin } \rightarrow \text{EXtea)} \) but not in LTL
satisfies $\Diamond \Box p$ but not $\text{AFAG}p$.  -- no single $F$ state

Similarly,

- $\Box \Diamond p \rightarrow \Diamond q$
- $\Diamond (p \land \neg p)$

not expressible in CTL
AGEFp - not expressible in LTL

Suppose $A\phi$ is equivalent to $\text{AGEFp}$
- (a) satisfies $\text{AGEFp}$ hence $A\phi$
- (b)-paths $\subseteq$ (a)-paths hence (b) satisfies $A\phi$
- However (b) does not satisfy $\text{AGEFp}$
CTL*

Allows any LTL formula to be preceded by A or E.

Exm.

\[ E(p \lor Xq), \ A(Fq \land Gp) \]

CTL* is more powerful than CTL and LTL but Model checking is PSPACE-complete in size of formula.
CTL Model Checking problem:

Given a program model \((S, s_0, R, L)\) and a CTL formula \(f\), determine if \(s_0 \models f\).

**Model-checking < Satisfiability**

Semantic search: Given, \(M=(S,s_0,R,L)\), every formula is identified with the set of states that satisfy the formula (false \(\rightarrow \emptyset\), true \(\rightarrow S\)). So, given \(f\),

- find the set \(S' \subseteq S\) of all states in \(S\) that satisfy \(f\).
- Check if \(s_0 \in S'\).
• Given \((S, s_0, R, L)\), and \(f\), find the set \(S' \subseteq S\) of all states in \(S\) that satisfy \(f\).

**Simple for formulae:**

\[
\{ s \mid s \models p \} = \{ s \mid p \in L(s) \} \quad \text{-- } p \text{ atomic}
\]

\[
\{ s \mid s \models f \land g \} = \{ s \mid s \models f \} \cap \{ s \mid s \models g \}
\]

\[
\{ s \mid s \models \neg f \} = S - \{ s \mid s \models f \}
\]

\[
\{ s \mid s \models \text{EX} f \} = \{ s \mid \exists t \in S \text{ s.t. } (s,t) \in R \land t \models f \}
\]

The problem arises with \(A(qUr)\) and \(E(qUr)\).
Fix-points

- Fix-point: $\tau(x)=x$

- Given a set $S$, a functional $\tau: 2^S \rightarrow 2^S$ is:
  - monotonic iff $P \subseteq Q \Rightarrow \tau[P] \subseteq \tau[Q]$
  - $\cup$-continuous iff $P_1 \subseteq P_2 \subseteq \ldots \Rightarrow \tau[\cup P_i]=\cup \tau[P_i]$
  - $\cap$-continuous iff $P_1 \supseteq P_2 \supseteq \ldots \Rightarrow \tau[\cap P_i]=\cap \tau[P_i]$

- Theorem (Tarski)

  Monotonic $\tau$ have least and greatest fix-points:
  
  \[
  \text{lfp}(\tau) = \bigcap \{Z : \tau[Z]=Z\},
  \]
  
  \[
  \text{gfp}(\tau) = \bigcup \{Z : \tau[Z]=Z\},
  \]

  \[
  \text{lfp}(\tau) = \bigcup \tau^i[\emptyset], \ i=0,1,\ldots \text{ if } \tau \text{ is also } \cup\text{-continuous}
  \]
  
  \[
  \text{gfp}(\tau) = \bigcap \tau^i[S], \ i=0,1,\ldots \text{ if } \tau \text{ is also } \cap\text{-continuous}
  \]
Lemma: $\text{EF}f = f \lor \text{EX}(\text{Eff})$ (satisfied by same states) 

$\Rightarrow$ $\text{EF}f$ is a fix-point of $\tau[Z] = f \lor \text{EX} Z$

Lemma: $f \lor \text{EX} Z$ is monotonic, $\cup$-continuous and $\cap$-continuous

$\Rightarrow$ by Tarski: $\text{lfp}(f \lor \text{EX} Z) = \lor(f \lor \text{EX} \text{false})^I$

Lemma: $\text{EF}f$ is a least fix-point of $\tau[Z] = f \lor \text{EX} Z$

$\Rightarrow \text{EF}f = \Box(f \square \text{EX} \text{false})^i$
## Fixpoints of CTL Operators (Clark-Emerson)

<table>
<thead>
<tr>
<th>$\tau[Z]$</th>
<th>lfp</th>
<th>computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g \lor (f \land \text{AXZ})$</td>
<td>$A[fUg]$</td>
<td>$\cup(g \lor (f \land \text{AX false}))^i$</td>
</tr>
<tr>
<td>$g \lor (f \land \text{EXZ})$</td>
<td>$E[fUg]$</td>
<td>$\cup(g \lor (f \land \text{EX false}))^i$</td>
</tr>
<tr>
<td>$f \lor \text{AXZ}$</td>
<td>$\text{AFF}f$</td>
<td>$\cup(f \lor \text{AX false})^i$</td>
</tr>
<tr>
<td>$f \lor \text{EXZ}$</td>
<td>$\text{EFF}f$</td>
<td>$\cup(f \lor \text{EX false})^i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\tau[Z]$</th>
<th>gfp</th>
<th>computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f \land \text{AXZ}$</td>
<td>$\text{AG}f$</td>
<td>$\cap(f \land \text{AX true})^i$</td>
</tr>
<tr>
<td>$f \land \text{EXZ}$</td>
<td>$\text{EG}f$</td>
<td>$\cap(f \land \text{EX true})^i$</td>
</tr>
</tbody>
</table>

where $i \leq \lvert S \rvert$ (S is finite)

Recall $\text{AX}(f) \equiv \neg \text{EX} \neg f$
Example: EFp

\[ \tau(\text{false}) = p \lor \text{EX } \text{false} = p \]

\[ \tau^2(\text{false}) = p \lor \text{EX } p \]

\[ \tau^3(\text{false}) = p \lor \text{EX}(p \lor \text{EX } p) \]

the Kripke model
Example: $EGy = \bigcap (y \land EX \text{ true})^i$

$[EG(y)] =$

$[y] \cap EX[S] = \{s_0,s_1,s_2,s_4\} = E_0$

$\cap [y] \cap EX[E_0] = \{s_0,s_1,s_2,s_4\} \cap \{s_0,s_2,s_3,s_4,s_5,s_6\} = \{s_0,s_2,s_4\} = E_1$

$\cap [y] \cap EX[E_1] = \{s_0,s_1,s_2,s_4\} \cap \{s_0,s_2,s_3,s_4,s_6\} = \{s_0,s_2,s_4\} = E_1$
Example: $\text{EF}(x=z \land y \neq z) = \bigcup((x=z \land y \neq z) \lor \text{EX} \text{false})^i$

$[\text{EF}(x=z \land y \neq z)] =$

$[(x=z \land y \neq z)] \cup \text{EX}[\emptyset] = \{s_4, s_5\} = \text{E}_0$

$\cup [\text{E}_0] \cup \text{EX}[\text{E}_0] = \{s_4, s_5\} \cup \{s_6\} = \{s_4, s_5, s_6\} = \text{E}_1$

$\cup (\text{EX}[\text{E}_1] = \{s_6, s_7\}) = \{s_4, s_5, s_6, s_7\} = \text{E}_2$

$\cup (\text{EX}[\text{E}_2] = \{s_6, s_7, s_5\}) = \{s_4, s_5, s_6, s_7\}$
Algorithm (sketch)
1. Construct A, the set of sub-formulae of f.
2. For $i=1,...,|f|$, label every state $s \in S$ with the sub-formulae of length $i$ that are satisfied by $s$.
3. Check if $s_0$ is labeled by f.

At stage $i$ the algorithm employs the information gathered in earlier stages (in particular path formulae look at the information of the next states).
Model Checking Algorithm

Given $M=(S, s_0, R, L)$, and a CTL formula $f$

for $j=0$ to $\text{length}(f)$

for each sub-formula $g$ of $f$ of length $j$

case (--- structure of $g$)

$p$: nothing  (--- $S$ is already labeled with propositions)
$q \land r$: $\forall s \in S$: if $q \in L(s)$ and $r \in L(s)$ then add $q \land r$ to $L(s)$
$\neg q$: $\forall s \in S$: if $q \notin L(s)$ then add $\neg q$ to $L(s)$
$\text{EX}q$: $\forall s \in S$: if $\exists t$ s.t. $(s,t) \in R$ and $q \in L(t)$ then add $\text{EX}q$ to $L(s)$
$A(q \lor r)$: $\text{AU-check}(q, r)$
$E(q \lor r)$: $\text{EU-check}(q, r)$

if $f \in L(s_0)$ then output=true, else output=false.
Checking state satisfiability for $A(q,Ur)$, $E(q,Ur)$

**AU-check** $(q, r)$

for each $s \in S$, if $r \in L(s)$ then add $A(q,Ur)$ to $L(s)$

for $j=1$ to $|S|$

for each $s \in S$,

if $q \in L(s)$ and $A(q,Ur) \in L(t)$ for all $t$ s.t. $(s,t) \in R$

then add $A(q,Ur)$ to $L(s)$

**EU-check** $(q, r)$

for each $s \in S$, if $r \in L(s)$ then add $E(q,Ur)$ to $L(s)$

for $j=1$ to $|S|$

for each $s \in S$,

if $q \in L(s)$ and $E(q,Ur)$ to $L(t)$ for some $t$ s.t. $(s,t) \in R$

then add $E(q,Ur)$ to $L(s)$

**Complexity**: $O(|\varphi| \cdot |S|^2)$
Fairness Assumption

Strong fairness: a transition that is enabled i.o. is taken i.o
- LTL: ($\square \diamond \varphi \rightarrow \square \diamond \psi$) - - $\square \diamond \psi$  private case, $\varphi=$True

Weak fairness: a transition that is continuously enabled is taken i.o
- LTL: ($\diamond \square \varphi \rightarrow \square \diamond \psi$)

$\Rightarrow$ Rule out traces that are unrealistic or provide for unfair service (starvation).

$\square \diamond (T_{Red} \& P_{Green})$
& $\square \diamond (P_{Red} \& T_{Green})$

Fairness assumptions are not in CTL
Fair Semantics for CTL

Strong Fairness constraint:

\[ S_{fair} = \bigwedge_i (\square \square \varphi_i \rightarrow \square \varphi_i) \land \bigwedge_j (\Diamond \square \varphi_j \rightarrow \square \Diamond \psi_j), \varphi_i, \psi_i, \varphi_j, \psi_j, \text{CTL formulae.} \]

For a model \( M = (S, s_I, R, L) \) and \( s \in S \):

\[ \text{FairPaths}(s) = \{ \pi \in \text{Paths}(s) \mid \pi \models S_{fair} \} \]

Fair semantics w.r.t. \( S_{fair} \)

\[ s \models_F p \quad \text{iff} \quad s \models p \]

\[ s \models_F \neg f \quad \text{iff} \quad \text{not } s \models_F f \]

\[ s \models_F f \land g \quad \text{iff} \quad s \models_F f \land s \models_F g \]

\[ s \models_F \exists x f \quad \text{iff} \quad \exists s'. (s, s') \in R \text{ and } s' \models f \land \pi \models \text{FairPaths}(s), \]

\[ s \models_F E[p \cup q] \quad \text{iff} \quad \exists \pi = (s_0, s_1, \ldots) \text{ s.t } s_0 = s, \pi \models \text{FairPaths}(s), \]

\[ \exists k \geq 0. s_k \models q \land \forall i. 0 \leq i < k \Rightarrow s_i \models p \]

\[ s \models_F A[p \cup q] \quad \text{iff} \quad \forall \pi = (s_0, s_1, \ldots) \text{ s.t } s_0 = s, \pi \models \text{FairPaths}(s), \]

\[ \exists k \geq 0. s_k \models q \land \forall i. 0 \leq i < k \Rightarrow s_i \models p \]
Model Checking under Farness Constraints

- How to check: \( \pi \in \text{FairPaths}(s) \) ?

- Observation: \( \pi \in \text{FairPaths}(s) \) iff \( \forall j \geq 0 \ \pi^j \in \text{FairPaths}(\pi_j) \)

- Hence, redefine fair semantic:

\[
\begin{align*}
    & s |=_F p & \text{iff} & s |=_p \\
    & s |=_F \neg f & \text{iff} & \neg s |=_F f \\
    & s |=_F f \land g & \text{iff} & s |=_F f \land s |=_F g \\
    & s |=_F \exists f & \text{iff} & \exists s'. (s,s') \in R \land s' |=_ f \land \text{FairPaths}(s') \neq \emptyset \\
    & s |=_F \forall f \text{U} g & \text{iff} & \text{for every path } (s_0, s_1, \ldots) \text{ s.t } s_0 = s, \\
    & & & \exists k \geq 0. s_k |=_ g \land \forall i. 0 \leq i < k \Rightarrow s_i |=_ f \\
    & & & \land \text{FairPaths}(s_k) \neq \emptyset \\
    & s |=_F \exists f \text{U} g & \text{iff} & \text{for some path } (s_0, s_1, \ldots) \text{ s.t } s_0 = s, \\
    & & & \exists k \geq 0. s_k |=_ g \land \forall i. 0 \leq i < k \Rightarrow s_i |=_ f \\
    & & & \land \text{FairPaths}(s_k) \neq \emptyset 
\end{align*}
\]
Computation of $\text{FairPaths}(s) \neq \emptyset$

- Let: $S_{fair} = (\Box \Diamond \varphi \rightarrow \Box \Diamond \psi)$ where $\varphi$, $\psi$ are CTL formulae.

  $\pi \models S_{fair}$ iff $\exists k \geq 0$, $n \geq k$, s.t. $\pi = s_0 \ldots s_{k-1}(s_k \ldots s_n)^\omega$

  and: $\forall k \leq i \leq n. s_i \not\models \varphi$ or $\exists k \leq j \leq n. s_j \models \psi$

- Computation of $\text{FairPaths}(s) \neq \emptyset$ w.r.t. $S_{fair} = (\Box \Diamond \varphi \rightarrow \Box \Diamond \psi)$

For a model $M = (S, s_I, R, L)$:

- Let $a, b, fair$ be new fresh variables

- Compute $Sat(\varphi) = \{ s \in S \mid s \models \varphi \}$ then $\forall s \in Sat(\varphi). L(s) := L(s) \cup \{a\}$

- Compute $Sat(\psi) = \{ s \in S \mid s \models \psi \}$ then $\forall s \in Sat(\psi). L(s) := L(s) \cup \{b\}$

- Decompose $M$ into a graph of maximal SCCs.

- For each SCC, if for all states $s \in \text{SCC}$, $a \not\in L(s)$, or otherwise there exists a state $s' \in \text{SCC}$ s.t. $b \not\in L(s')$, mark the SCC as $fair$

- For each $s \in S$ if there exists a path from $s$ to a $fair$ SCC then $L(s) \cup \{fair\}$
Computation of $\text{FairPaths}(s) \neq \emptyset$

- Hence, redefine fair semantic:

  \[
  s \models_F p \quad \text{iff} \quad s \models p
  \]

  \[
  s \models_F \neg f \quad \text{iff} \quad \neg s \models_F f
  \]

  \[
  s \models_F f \land g \quad \text{iff} \quad s \models_F f \land s \models_F g
  \]

  \[
  s \models_F \text{E}_f \quad \text{iff} \quad \exists s'. (s, s') \in R \land s' \models (f \land \text{fair})
  \]

  \[
  s \models_F \text{A}[f \land g] \quad \text{iff} \quad \text{for every path } (s_0, s_1, \ldots) \text{ s.t } s_0 = s, \exists k \geq 0. s_k \models g \land \forall i. 0 \leq i < k \Rightarrow s_i \models (f \land \text{fair})
  \]

  \[
  s \models_F \text{E}[f \land g] \quad \text{iff} \quad \text{for some path } (s_0, s_1, \ldots) \text{ s.t } s_0 = s, \exists k \geq 0. s_k \models g \land \forall i. 0 \leq i < k \Rightarrow s_i \models (f \land \text{fair})
  \]

Thus, ‘fair’ model checking requires pre-processing of the model that extends the states labeling, then normal model checking algorithm.
• For a model with $N$ states and $M$ transitions and a CTL formula $f$ and a CTL fairness constraint, still linear time: $O(|f| \cdot (N + M))$.

• In the general case, with $k$ fairness constraints, the labeling must be carried out separately for each constraint and the final labeling is the conjunction of the separate labeling. The complexity is then $O(|f| \cdot (N + M) \cdot k)$.
A(qU^{\leq k} r) - at all paths r holds within k t.u. and q holds at all states until then.

E(qU^{\leq k} r) - at some path r holds within k t.u. and q holds at all states until then.

Semantics (fixed rate progress approach):

\( s_0 \models A(qU^{\leq k} r) \) iff for every path \(( s_0, s_1, \ldots)\)

\( \exists i. \ 0 \leq i \leq k \land s_i \models r \land \forall j. \ 0 \leq j < i \Rightarrow s_j \models q \)

\( s_0 \models E(qU^{\leq k} r) \) iff exists path \(( s_0, s_1, \ldots)\) and

\( \exists i \geq 0. \ 0 \leq i \leq k \land s_i \models r \land \forall j. \ 0 \leq j < i \Rightarrow s_j \models q \)
Given $M=(S, s_0, R, L)$, and a CTL formula $f$

for $j=0$ to $\text{length}(f)$

for each sub-formula $g$ of $f$ of length $j$

case (structure of $g$)

$p$: 

$q \land r$: 

$\neg q$: 

$\text{EX}q$: 

$\text{A}(qUr)$: 

$\text{A}(qU^{\leq k}r)$: $\text{AU-check}(\text{min}(k,|S|), q, r)$

$\text{E}(qUr)$: 

$\text{E}(qU^{\leq k}r)$: $\text{EU-check}(\text{min}(k,|S|), q, r)$

if $f \in L(s_0)$ then output=true, else output=false.
A(qU≤kr):

AU-check (max_len, q, r) -- max_len= \text{min}(k, |S|)

for each \( s \in S \), if \( r \in L(s) \) then add \( A(qU^{\leq 0}r) \) to \( L(s) \)

for \( len=1 \) to \( \text{max}_\text{len} \)

for each \( s \in S \),

if \( q \in L(s) \) and for every \( t \) such that \( (s, t) \in R \)

there is \( j \leq len-1 \) s.t. \( A(qU^{\leq j}r) \in L(t) \)

then add \( A(qU^{\leq len}r) \) to \( L(s) \)

for each \( s \in S \),

if \( A(qU^{\leq j}r) \in L(s) \) for some \( j \leq \text{max}_\text{len} \)

then replace by \( A(qU^{\leq \text{max}_\text{len}}r) \) to \( L(s) \)

Note \( j<\text{max}_\text{len} \) can occur when \( |S|>k \)
\textbf{EU-check (max\_len, q, r)} -- \textit{max\_len} = \textit{min}(k, |S|)

\begin{enumerate}
\item for each \( s \in S \), if \( r \in L(s) \) then add \( E(qU^{\leq 0}r) \) to \( L(s) \)
\item for \( \text{len}=1 \) to \( \text{max\_len} \)
\begin{enumerate}
\item for each \( s \in S \),
\begin{enumerate}
\item if \( q \in L(s) \) and
\begin{enumerate}
\item \( E(qU^{\leq \text{len}-1}r) \in L(t) \) for some \( t \) such that \( (s,t) \in R \)
\end{enumerate}
\end{enumerate}
then add \( E(qU^{\leq \text{len}}r) \) to \( L(s) \)
\item for each \( s \in S \),
\begin{enumerate}
\item if \( E(qU^{\leq j}r) \in L(s) \) for some \( j \leq \text{max\_len} \)
\begin{enumerate}
\item then replace by \( E(qU^{\leq \text{max\_len}}r) \) to \( L(s) \)
\end{enumerate}
\end{enumerate}
\end{enumerate}
\end{enumerate}
\end{enumerate}
In order to express fixed delays, we need:

\[ A(qU^k r) - \text{at all paths } r \text{ holds at } k \text{ t.u. and } q \text{ holds at all states until then.} \]

\[ E(qU^k r) - \text{at some path } r \text{ holds at } k \text{ t.u. and } q \text{ holds at all states until then.} \]

Semantics (fixed rate progress approach):

\[ s_0 \models A(qU^k r) \iff \text{for every path } (s_0, s_1, \ldots) \]
\[ s_k \models r \land \forall j. \ 0 \leq j < k \Rightarrow s_j \models q \]

\[ s_0 \models E(qU^k r) \iff \text{exists path } (s_0, s_1, \ldots) \text{ and} \]
\[ s_k \models r \land \forall j. \ 0 \leq j < k \Rightarrow s_j \models q \]

Model checking same idea as for \( \leq k \)
Railroad Crossing in Real-Time CTL

Assertions:

• 50 seconds minimal delay between trains.
  \[ \text{AG}(\text{Tin} \Rightarrow \neg \text{EF}^{\leq 50}\text{Tin}) \]

• It takes a train 6 seconds to arrive at the signal.
  \[ \text{AG}(\text{Tin} \Rightarrow \text{AF}^6\text{AtSignal}) \]

• Train exits XR within 15 to 25 seconds.
  \[ \text{AG}((((\text{Twait} \land \text{AX}(\neg \text{Twait})) \lor \text{AX}((\text{AtSignal} \land \neg \text{Twait}))) \Rightarrow \text{AG}^{\leq 16} \neg \text{Tout} \land \text{AF}^{\leq 26}\text{Tout})) \]

Requirements:

• Train at the signal is allowed to continue within 10 seconds.
  \[ \text{AG}((\text{AtSignal} \Rightarrow \text{AF}^{\leq 5}(\neg \text{Twait}))) \]

• The gate is open whenever the crossing is empty for more than 10 seconds.
  \[ \text{AG}(\neg \text{E}(\text{Tcr}^{0}\text{U}=^{10}\neg \text{Open})) \]
Symbolic Model Checking

Symbolic representation with Boolean formulae of:

- the model (program)
- CTL formula

thus avoiding the state explosion problem.

∴ Model Checking reduces to Boolean formulae manipulation.

The idea: Given a model \((S,R,L)\) over \(P\) (set of atomic propositions),

- Represent every state \(s \in S\) by a Boolean formula over \(P\) that is satisfied exactly by \(L(s)\)
- Represents \(R\) in terms of the Boolean formulas corresponding to source and destination states.
Program Representation - Example I

\[ S \equiv \neg p \]
\[ S' \equiv p \]
\[ R \equiv (\neg p \land p') \lor (p \land p') \lor (p \land \neg p') \]
\[ \equiv (\neg p \lor p) \land p' \lor (p \land \neg p') \]
\[ \equiv p' \lor (p \land \neg p') \]
\[ \equiv (p' \lor p) \land (p' \lor \neg p') \]
\[ \equiv (p \lor p') \]
S0 ≡ \neg(v_0 \lor v_1), \quad S1 ≡ v_0 \land \neg v_1,
S2 ≡ \neg v_0 \land v_1, \quad S3 ≡ v_0 \land v_1
R \equiv (v_0' \leftrightarrow \neg v_0) \land (v_1' \leftrightarrow (v_0 \oplus v_1))
CTL Symbolic Representation

- \( p \in P \) , \( \neg f \) , \( f \land g \) , are Boolean formulae.
- We show Boolean representation for: \( \text{EX}f \) , \( \text{A}[f \lor g] \) , \( \text{E}[f \lor g] \)

**EX\( f \)**

- Let \( v_1,...,v_n \in P \) be the atomic propositions in \( f \), then:

  \[
  \exists (v_1,...,v_n).f = \lor (a_1,...,a_n) \in \{0,1\}^n \ f(a_1,...,a_n)
  \]

- Then, \( \text{EX}f \) is represented by the formula

  \[
  \exists (v_1',...,v_n').(R \land f'(v_1',...,v_n'))
  \]

**\( \text{EX}\neg p \)**

\[
\text{EX}\neg p = \exists p'. R \land f(p')
\]
\[
= \exists p'. (p \lor p') \land \neg p'
\]
\[
= \exists p'. (p \land \neg p')
\]
\[
= (p \land \neg \text{false}) \lor (p \land \neg \text{true}) = p
\]
EX(v₀ ∧ v₁)
= ∃(v₀', v₁'). R ∧ (v₀' ∧ v₁')
= ∃(v₀', v₁'). (¬v₀ ∧ v₁ ∧ v₀' ∧ v₁')
= ∃(v₁'). (¬v₀ ∧ v₁ ∧ false ∧ v₁') ∨ (¬v₀ ∧ v₁ ∧ true ∧ v₁')
= ∃(v₁'). (¬v₀ ∧ v₁ ∧ v₁')
= (¬v₀ ∧ v₁ ∧ false) ∨ (¬v₀ ∧ v₁ ∧ true)
= (¬v₀ ∧ v₁)
For other CTL operators use their fixed point representation in terms of EX.

Example: computation of EFp
- recall EFp = ∪τ^i(false) where τ(y)= p ∨ EXy
  τ^1(false) = p ∨ EXfalse = p
  τ^2(false) = τ(p) = p ∨ EXp = p ∨ ∃p'.(p ∨ p') ∧ p' = true
  τ^3(false) = τ(true) = p ∨ EXtrue = true
Computation of $\text{EF}(v_0 \land v_1)$

\[ \tau^1(\text{false}) = (v_0 \land v_1) \lor \text{EXfalse} = (v_0 \land v_1) \]

\[ \tau^2(\text{false}) = \tau(v_0 \land v_1) = (v_0 \land v_1) \lor \text{EX}(v_0 \land v_1) = (v_0 \land v_1) \lor (\neg v_0 \land v_1) = v_1 \]

\[ \tau^3(\text{false}) = \tau(v_1) = (v_0 \land v_1) \lor \text{EX}v_1 = (v_0 \lor v_1) \]

\[ \tau^3(\text{false}) = \tau(v_0 \lor v_1) = (v_0 \land v_1) \lor \text{EX}(v_0 \lor v_1) = \text{true} \]
Symbolic Model Checking Algorithm

eval(f)

{ Case (f)
  p : return p
  \neg g: return \neg eval(g)
  g \lor h: return eval(g) \lor eval(h)
  EXg: return \exists v'.(R \land p')
  E(gU h): return evalEU(eval(g),eval(h),false))
  EGg: return evalEGG(eval(g),true) }
Graphical representation of Boolean formulae obtained from ordered decision trees.

Ordered Decision Tree of $a \land b \lor c \land d$
Apply in bottom-up manner:

- Combine isomorphic sub-trees into a single tree
- Eliminate nodes whose left and right children are isomorphic
  - Linear time
  - Size of resulting graph depends on variable ordering
Example - Step 1: node elimination
Example - Step 1: isomorphic trees
Example - Step 1: Final BDD
Properties of BDDs

• Canonical representation of formulae, enables simple comparison.
• Logical operators: $\neg$, $\land$, $\lor$, EX are computed w.r.t. BDD quadratic time.