Computation Tree Logic

Luis Tertulino & Ronaldo Silveira

October 23, 2015

- A brief introduction to Propositional Logic, its syntax and its semantics

- A brief introduction to Propositional Logic, its syntax and its semantics
- Formal models of time

- A brief introduction to Propositional Logic, its syntax and its semantics
- Formal models of time
  - Frames and Flows of time

- A brief introduction to Propositional Logic, its syntax and its semantics
- Formal models of time
  - Frames and Flows of time
- Temporal Logic extends the Propositional Logic
  - The connectives H and G

- A brief introduction to Propositional Logic, its syntax and its semantics
- Formal models of time
  - Frames and Flows of time
- Temporal Logic extends the Propositional Logic
  - The connectives H and G
- Some practical applications

- "What good is Temporal Logic?"

- "What good is Temporal Logic?"
  - Answer: "Temporal Logic is a good method for specifying and reasoning about a concurrent program".

- “What good is Temporal Logic?”
  - Answer: “Temporal Logic is a good method for specifying and reasoning about a concurrent program”.
- Some mathematical definitions about states of a program and (concurrent) programs

- "What good is Temporal Logic?"
  - Answer: "Temporal Logic is a good method for specifying and reasoning about a concurrent program".
- Some mathematical definitions about states of a program and (concurrent) programs
- The Peterson's algorithm

- "What good is Temporal Logic?"
  - Answer: "Temporal Logic is a good method for specifying and reasoning about a concurrent program".
- Some mathematical definitions about states of a program and (concurrent) programs
- The Peterson's algorithm
- How the model checking works?

- "What good is Temporal Logic?"
  - Answer: "Temporal Logic is a good method for specifying and reasoning about a concurrent program".
- Some mathematical definitions about states of a program and (concurrent) programs
- The Peterson's algorithm
- How the model checking works?
- Some strengths and weaknesses of model checking

- Syntax and Semantics of LTL

Previously on Temporal Logic Week...

Linear Temporal Logic

- Syntax and Semantics of LTL
- ω-languages, Kripke structures, paths and traces
- Buchi automata and LTL model checking

- Needing of expressing uncertainty;

Motivation

- Needing of expressing uncertainty;
- Different paths of the future;

Intuition

In Computation Tree Logic (CTL) the model of time is a tree-like structure. This way, we cannot use Linear Temporal Logic (LTL) to express the existence of a certain path of time in which some event occurs.

The syntax of CTL consists on the syntax of temporal logic plus some path operators. The class of formulas can be defined in Backus-Naur form. If ϕ is a formula:

The syntax of CTL consists on the syntax of temporal logic plus some path operators. The class of formulas can be defined in Backus-Naur form. If ϕ is a formula:

$$\phi ::= \bot \mid \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid AX\phi \mid EX\phi \mid$$

$$AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A[\phi U \phi] \mid E[\phi U \phi]$$

The syntax of CTL consists on the syntax of temporal logic plus some path operators. The class of formulas can be defined in Backus-Naur form. If ϕ is a formula:

$$\phi ::= \bot \mid \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid AX\phi \mid EX\phi \mid$$

$$AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A[\phi U\phi] \mid E[\phi U\phi]$$

With p as a literal (atomic formula), AX, EX, AF, EF, AG e EG unary operators.

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.
The path-specific operators can be read as:

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.
The path-specific operators can be read as:

- A: is the universal quantifier over paths. Read as: "in all possible paths";

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- A: is the universal quantifier over paths. Read as: "in all possible paths";
- E:is the existential quantifier over paths. Read as: "exists a path in which";

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.
The path-specific operators can be read as:

- A: is the universal quantifier over paths. Read as: "in all possible paths";
- E:is the existential quantifier over paths. Read as: "exists a path in which";

The temporal operators, as in LTL, can be read as:

- X: "in the next state";

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- *A*: is the universal quantifier over paths. Read as: "in all possible paths";
- *E*:is the existential quantifier over paths. Read as: "exists a path in which";

The temporal operators, as in LTL, can be read as:

- *X*: "in the next state";
- *F* "There is some state in the future (eventually)";

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.

The path-specific operators can be read as:

- *A*: is the universal quantifier over paths. Read as: "in all possible paths";
- *E*:is the existential quantifier over paths. Read as: "exists a path in which";

The temporal operators, as in LTL, can be read as:

- *X*: "in the next state";
- *F* "There is some state in the future (eventually)";
- *G* "Globally (in all future states)";

The propositional operators: $\neg, \vee, \wedge, \rightarrow$ have the same meaning of in the propositional logic.
The path-specific operators can be read as:

- *A*: is the universal quantifier over paths. Read as: "in all possible paths";
- *E*:is the existential quantifier over paths. Read as: "exists a path in which";

The temporal operators, as in LTL, can be read as:

- *X*: "in the next state";
- *F* "There is some state in the future (eventually)";
- *G* "Globally (in all future states)";
- $\varphi U \psi$: φ is true at least until ψ becomes true;

- Notice that, in CTL, the combination of path specific operators and temporal operators are atomic, e.g., AF is an atomic operator that can be read as "In all paths in the future there is some state where..";

- Notice that, in CTL, the combination of path specific operators and temporal operators are atomic, e.g., AF is an atomic operator that can be read as "In all paths in the future there is some state where...";
- Notice as well that the binary operators $A[\varphi U \psi]$ and $E[\varphi U \psi]$ can be represented as AU and EU, respectively;

- Notice that, in CTL, the combination of path specific operators and temporal operators are atomic, e.g., AF is an atomic operator that can be read as "In all paths in the future there is some state where..";
- Notice as well that the binary operators $A[\varphi U \psi]$ and $E[\varphi U \psi]$ can be represented as AU and EU, respectively;
- We assume that, similarly to the $\neg$ operator, the "new" unary operators (AX, EX, AF, EF, AG, and EG) have the first precedence. Next comes the $\wedge$ and $\vee$ operators. And at last the $\rightarrow$, AU and EU;

Examples

- Examples of well-formed formulas:
  - $AG(p \vee EFq)$

- Examples of well-formed formulas:
  - $AG(p \vee EFq)$
  - $AX(q \rightarrow E[(p \vee q)Ur])$

Examples

- Examples of well-formed formulas:
  - $AG(p \vee EFq)$
  - $AX(q \rightarrow E[(p \vee q)Ur])$
  - $EFEGp \rightarrow AFr$ Note that this is binded as $(EFEGp) \rightarrow AFr$, not as $EFEG(p \rightarrow AFr)$

Examples

- Examples of well-formed formulas:
  - $AG(p \vee EFq)$
  - $AX(q \rightarrow E[(p \vee q)Ur])$
  - $EFEGp \rightarrow AFr$ Note that this is binded as $(EFEGp) \rightarrow AFr$, not as $EFEG(p \rightarrow AFr)$
- Example of formulas that are not well-formed:
  - $A\neg G\neg p$

Examples

- Examples of well-formed formulas:
  - $AG(p \vee EFq)$
  - $AX(q \to E[(p \vee q)Ur])$
  - $EFEGp \to AFr$ Note that this is binded as $(EFEGp) \to AFr$, not as $EFEG(p \to AFr)$
- Example of formulas that are not well-formed:
  - $A\neg G\neg p$
  - $F[pUs]$

Examples

- Examples of well-formed formulas:
  - $AG(p \vee EFq)$
  - $AX(q \rightarrow E[(p \vee q)Ur])$
  - $EFEGp \rightarrow AFr$ Note that this is binded as $(EFEGp) \rightarrow AFr$, not as $EFEG(p \rightarrow AFr)$
- Example of formulas that are not well-formed:
  - $A\neg G\neg p$
  - $F[pUs]$
  - $A[pUs \wedge qUs]$

Different from usual logics, CTL formulas are interpreted by a transition system. Given an set of atoms:

Different from usual logics, CTL formulas are interpreted by a transition system. Given an set of atoms:

Definition (1)

A **transition system** $\mathcal{M}$ is a triple $\mathcal{M} = (S, \rightarrow, L)$ in which S is a set of states, $\rightarrow$ is a binary relation over S ($\rightarrow \subseteq S \times S$) and $L : S \rightarrow \mathcal{P}(Atoms)$ is a labelling function.

Different from usual logics, CTL formulas are interpreted by a transition system. Given an set of atoms:

Definition (1)

A **transition system** $\mathcal{M}$ is a triple $\mathcal{M} = (S, \rightarrow, L)$ in which S is a set of states, $\rightarrow$ is a binary relation over S ($\rightarrow \subseteq S \times S$) and $L : S \rightarrow \mathcal{P}(Atoms)$ is a labelling function.

Definition (2)

A **model** is a duple $\mathcal{M}, s$ in which $\mathcal{M}$ is a transition system and $s \in S$ is a state of the transition system.

Different from usual logics, CTL formulas are interpreted by a transition system. Given an set of atoms:

Definition (1)

A **transition system** $\mathcal{M}$ is a triple $\mathcal{M} = (S, \rightarrow, L)$ in which S is a set of states, $\rightarrow$ is a binary relation over S ($\rightarrow \subseteq S \times S$) and $L : S \rightarrow \mathcal{P}(Atoms)$ is a labelling function.

Definition (2)

A **model** is a duple $\mathcal{M}, s$ in which $\mathcal{M}$ is a transition system and $s \in S$ is a state of the transition system.

Notation: we will use $\mathcal{M}, s \vDash \varphi$ to denote that the model $\mathcal{M}, s$ satisfies the formula φ

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash \top$ and $\mathcal{M}, s \nvDash \bot$ for all $s \in S$

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \models \top$ and $\mathcal{M}, s \not\models \bot$ for all $s \in S$
- $\mathcal{M}, s \models p$ iff $p \in L(S)$

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash \top$ and $\mathcal{M}, s \nvDash \bot$ for all $s \in S$
- $\mathcal{M}, s \vDash p$ iff $p \in L(S)$
- $\mathcal{M}, s \vDash \neg\varphi$ iff $\mathcal{M}, s \nvDash \varphi$

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash \top$ and $\mathcal{M}, s \nvDash \bot$ for all $s \in S$
- $\mathcal{M}, s \vDash p$ iff $p \in L(S)$
- $\mathcal{M}, s \vDash \neg\varphi$ iff $\mathcal{M}, s \nvDash \varphi$
- $\mathcal{M}, s \vDash \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, s \vDash \varphi_1$ AND $\mathcal{M}, s \vDash \varphi_2$

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash \top$ and $\mathcal{M}, s \nvDash \bot$ for all $s \in S$
- $\mathcal{M}, s \vDash p$ iff $p \in L(S)$
- $\mathcal{M}, s \vDash \neg\varphi$ iff $\mathcal{M}, s \nvDash \varphi$
- $\mathcal{M}, s \vDash \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, s \vDash \varphi_1$ AND $\mathcal{M}, s \vDash \varphi_2$
- $\mathcal{M}, s \vDash \varphi_1 \vee \varphi_2$ iff $\mathcal{M}, s \vDash \varphi_1$ OR $\mathcal{M}, s \vDash \varphi_2$

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash \top$ and $\mathcal{M}, s \nvDash \bot$ for all $s \in S$
- $\mathcal{M}, s \vDash p$ iff $p \in L(S)$
- $\mathcal{M}, s \vDash \neg\varphi$ iff $\mathcal{M}, s \nvDash \varphi$
- $\mathcal{M}, s \vDash \varphi_1 \wedge \varphi_2$ iff $\mathcal{M}, s \vDash \varphi_1$ AND $\mathcal{M}, s \vDash \varphi_2$
- $\mathcal{M}, s \vDash \varphi_1 \vee \varphi_2$ iff $\mathcal{M}, s \vDash \varphi_1$ OR $\mathcal{M}, s \vDash \varphi_2$
- $\mathcal{M}, s \vDash \varphi_1 \rightarrow \varphi_2$ iff $\mathcal{M}, s \nvDash \varphi_1$ OR $\mathcal{M}, s \vDash \varphi_2$

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash AX\varphi$ iff for all s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \vDash \varphi$. Thus, AX says: "in every next state..."

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash AX\varphi$ iff for all s_1 that $s \to s_1$ and $\mathcal{M}, s_1 \vDash \varphi$. Thus, AX says: "in every next state..."
- $\mathcal{M}, s \vDash EX\varphi$ iff exists s_1 that $s \to s_1$ and $M, s_1 \vDash \varphi$. Thus, EX says: "in some next state..."

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash AX\varphi$ iff for all s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \vDash \varphi$. Thus, AX says: "in every next state..."
- $\mathcal{M}, s \vDash EX\varphi$ iff exists s_1 that $s \rightarrow s_1$ and $M, s_1 \vDash \varphi$. Thus, EX says: "in some next state..."
- $\mathcal{M}, s, \vDash AG\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ...$ in which $s = s_1$, for all s_i, $\mathcal{M}, s_i \vDash \varphi$. Thus, AG says: "In all possible paths from now on in all next states..."

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s \vDash AX\varphi$ iff for all s_1 that $s \rightarrow s_1$ and $\mathcal{M}, s_1 \vDash \varphi$. Thus, AX says: "in every next state..."
- $\mathcal{M}, s \vDash EX\varphi$ iff exists s_1 that $s \rightarrow s_1$ and $M, s_1 \vDash \varphi$. Thus, EX says: "in some next state..."
- $\mathcal{M}, s, \vDash AG\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ in which $s = s_1$, for all s_i, $\mathcal{M}, s_i \vDash \varphi$. Thus, AG says: "In all possible paths from now on in all next states..."
- $\mathcal{M}, s, \vDash AG\varphi$ iff exists some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ in which $s = s_1$, for all s_i, $\mathcal{M}, s_i \vDash \varphi$ Thus, EG says: "Exists a path from now on in all next states..."

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s, \models AF\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ...$ in which $s = s_1$, exists s_i, $\mathcal{M}, s_i \models \varphi$. Thus, AF says: "In all possible paths from now on, in some next state..."

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s, \vDash AF\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ...$ in which $s = s_1$, exists s_i, $\mathcal{M}, s_i \vDash \varphi$. Thus, AF says: "In all possible paths from now on, in some next state..."
- $\mathcal{M}, s, \vDash EF\varphi$ iff exists some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ...$ in which $s = s_1$, that exists s_i, $\mathcal{M}, s_i \vDash \varphi$. Thus, EF says: "In some path from now on, in some next state..."

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s, \vDash AF\varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ...$ in which $s = s_1$, exists s_i, $\mathcal{M}, s_i \vDash \varphi$. Thus, AF says: "In all possible paths from now on, in some next state..."
- $\mathcal{M}, s, \vDash EF\varphi$ iff exists some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ...$ in which $s = s_1$, that exists s_i, $\mathcal{M}, s_i \vDash \varphi$. Thus, EF says: "In some path from now on, in some next state..."
- $\mathcal{M}, s, \vDash A[\varphi_1 U \varphi_2]$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ...$ in which $s = s_1$, this path satisfies $\varphi_1 U \varphi_2$, i.e., exists s_i in the path such that $\mathcal{M}, s_i \vDash \varphi_2$ and, for all $j < i$, $\mathcal{M}, s_j \vDash \varphi_1$. Thus, AU says: "For all paths from now on, until some state..."

Take an arbitrary model $\mathcal{M}$. Let s, s_1, s_2, s_3 be states in S. Let $\varphi, \varphi_1, \varphi_2$ be well-formed formulas of CTL. And let p be an atom. The satisfaction of CTL formulas can be defined as follows:

- $\mathcal{M}, s, \vDash E[\varphi_1 U \varphi_2]$ iff exists some path $s_1 \to s_2 \to s_3 \to ...$ in which $s = s_1$, this path satisfies $\varphi_1 U \varphi_2$. Thus, EU says: "In some path from now on, until some state..."

- "It's possible to get to a state where something has started but it's not ready": $EF(started \land \neg ready)$

Examples

- "It's possible to get to a state where something has started but it's not ready": $EF(started \land \neg ready)$
- "A certain process is enabled infinitely often on every computation path": $AG(AFenabled)$

- "It's possible to get to a state where something has started but it's not ready": $EF(started \land \neg ready)$
- "A certain process is enabled infinitely often on every computation path": $AG(AFenabled)$
- "An upwards travelling lift at the second floor does not change its direction when it has passengers wishing to go to the fifth floor":
$AG(floor2 \land directionup \land button5 \rightarrow A[directionup U floor5])$

Equivalences

Definition

Two CTL formulas φ and ψ are said to be **semantically equivalent** if any state in any model which satisfies one of them also satisfies the other.

Definition

Two CTL formulas φ and ψ are said to be **semantically equivalent** if any state in any model which satisfies one of them also satisfies the other.

Notation: we denote the semantic equivalence of φ and ψ by $\varphi \equiv \psi$

Example of equivalences

Let φ be an arbitrary CTL formula.

- $\neg AF\varphi \equiv EG\neg\varphi$

Let φ be an arbitrary CTL formula.

- $\neg AF\varphi \equiv EG\neg\varphi$
- $\neg EF\varphi \equiv AG\neg\varphi$

Example of equivalences

Let φ be an arbitrary CTL formula.

- $\neg AF\varphi \equiv EG\neg\varphi$
- $\neg EF\varphi \equiv AG\neg\varphi$
- $\neg AX\varphi \equiv EX\neg\varphi$

Example of equivalences

Let φ be an arbitrary CTL formula.

- $\neg AF\varphi \equiv EG\neg\varphi$
- $\neg EF\varphi \equiv AG\neg\varphi$
- $\neg AX\varphi \equiv EX\neg\varphi$
- $AF\varphi \equiv A[\top U\varphi]$

Example of equivalences

Let φ be an arbitrary CTL formula.

- $\neg AF\varphi \equiv EG\neg\varphi$
- $\neg EF\varphi \equiv AG\neg\varphi$
- $\neg AX\varphi \equiv EX\neg\varphi$
- $AF\varphi \equiv A[\top U\varphi]$
- $EF\varphi \equiv E[\top U\varphi]$

Because of the equivalences shown and the ones in propositional logic, we can have some minimum sets of conectives for the CTL syntax. One of them is defined in Backus-Naur formalism below:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \rightarrow \phi \mid AX\phi \mid A[\phi U\phi] \mid E[\phi U\phi]$$

Even if CTL allow explicit quantification over paths, it cannot allow some expressions to be formed. For example, we cannot say, as in LTL: "All paths in which have p on them, also have q on them".
This expression can be translated in LTL as follows:

$$Fp \to Fq$$

We can try expressing it as $AFp \rightarrow AFq$ but it does not have the same meaning. This one statement means "If all paths have a p along them, then all paths have a q along then"

We can try expressing it as $AFp \rightarrow AFq$ but it does not have the same meaning. This one statement means "If all paths have a p along them, then all paths have a q along then"
We can try to translate it as $AG(p \rightarrow AFq)$ which is closer, but not exactly the same. This one means "for all paths, in all states on the future, if they hold p then, all paths will eventualy hold q"

For this, we can extend the CTL by dropping the constraint that every temporal operator (X, U, F, G) has to be associated with an unique path quantifier (A, E).

For this, we can extend the CTL by dropping the constraint that every temporal operator (X, U, F, G) has to be associated with an unique path quantifier (A, E).
This allows us to generate some statements:

- "In all possible paths, q is true until r is true or p is true until r is true": $A[qUr \vee pUr]$

- "In all possible paths, q is true until r is true or p is true until r is true": $A[qUr \vee pUr]$
- "There is a path in which p eventually occurring will occur in all states": $E[GFp]$

- "In all possible paths, q is true until r is true or p is true until r is true": $A[qUr \vee pUr]$
- "There is a path in which p eventually occurring will occur in all states": $E[GFp]$
- "In all paths, p will occur in the next state or in the next of the next": $A[Xp \vee XXp]$

The syntax of CTL* can be defined with the BNF bellow:

$$\phi ::= \bot \mid \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid A[\alpha] \mid E[\alpha] \mid$$

$$\alpha ::= \phi \mid \neg\alpha \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \alpha \rightarrow \alpha \mid \alpha U \alpha \mid G\alpha \mid F\alpha \mid X\alpha \mid$$

With the same meanings of each operator.

Although we don't define path operators to LTL we can assume that it consider in all paths. Therefore, we can say that a formula ϕ in LTL is a formula $A[\phi]$ in CTL*;

Although we don't define path operators to LTL we can assume that it consider in all paths. Therefore, we can say that a formula ϕ in LTL is a formula $A[\phi]$ in CTL*;
For CTL, it is trivial;