Static Analysis and Verification

SAV 2024/2025

Tomáš Vojnar

vojnar@fit.vutbr.cz

Brno University of Technology
Faculty of Information Technology
Božetěchova 2, 612 66 Brno

Temporal Logics:

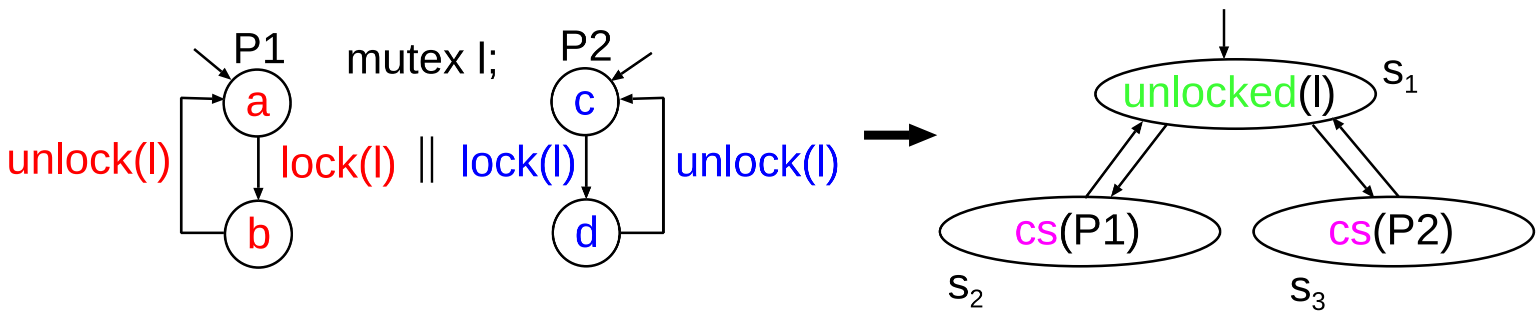
CTL*, CTL, LTL

Model of Computation

Kripke Structures

❖ Informally, Kripke structures are directed graphs whose

- vertices correspond to configurations of the examined system,
- the vertices are labelled by atomic propositions that are true in the appropriate configurations, and
- edges encode possible transitions between the configurations.



❖ Can be generated from the source description of examined systems (or used implicitly as an underlying semantic model of the formulae as well as examined systems).

❖ The generation involves the state explosion problem, or the Kripke structure may be infinite—in the following, we, however, concentrate on finite Kripke structures.

Kripke Structures

❖ Let AP be a set of atomic propositions about the configurations of the examined system.

❖ Formally, a (finite) Kripke structure M over AP is a tuple $M = (S, S_0, R, L)$ where

- S is a finite set of states,
- $S_0 \subseteq S$ is a set of initial states,
- $R \subseteq S \times S$ is a transition relation, for convenience supposed to be total (i.e. $\forall s \in S\ \exists s' \in S.\ R(s, s')$),
- $L : S \rightarrow 2^{AP}$ is a labelling function that labels each state by the set of atomic propositions that are true in it.

Kripke Structures

- Let AP be a set of atomic propositions about the configurations of the examined system.
- Formally, a (finite) Kripke structure M over AP is a tuple $M = (S, S_0, R, L)$ where
  - S is a finite set of states,
  - $S_0 \subseteq S$ is a set of initial states,
  - $R \subseteq S \times S$ is a transition relation, for convenience supposed to be total (i.e. $\forall s \in S \; \exists s' \in S.\; R(s, s')$),
  - $L : S \rightarrow 2^{AP}$ is a labelling function that labels each state by the set of atomic propositions that are true in it.
- For the example from the previous slide, we have:
  - $AP = \{unlocked(l), cs(P1), cs(P2)\}$,
  - $S = \{s_1, s_2, s_3\}$,
  - $S_0 = \{s_1\}$,
  - $R = \{(s_1, s_2), (s_2, s_1), (s_1, s_3), (s_3, s_1)\}$,
  - $L = \{(s_1, \{unlocked(l)\}), (s_2, \{cs(P1)\}), (s_3, \{cs(P2)\})\}$.

Kripke Structures

- A path π in a Kripke structure M is an infinite sequence of states $\pi = s_0 s_1 s_2 ...$ such that $\forall i \in \mathbb{N}. R(s_i, s_{i+1})$.

- We denote $\Pi(M, s)$ the set of all paths in M that start at $s \in S$.

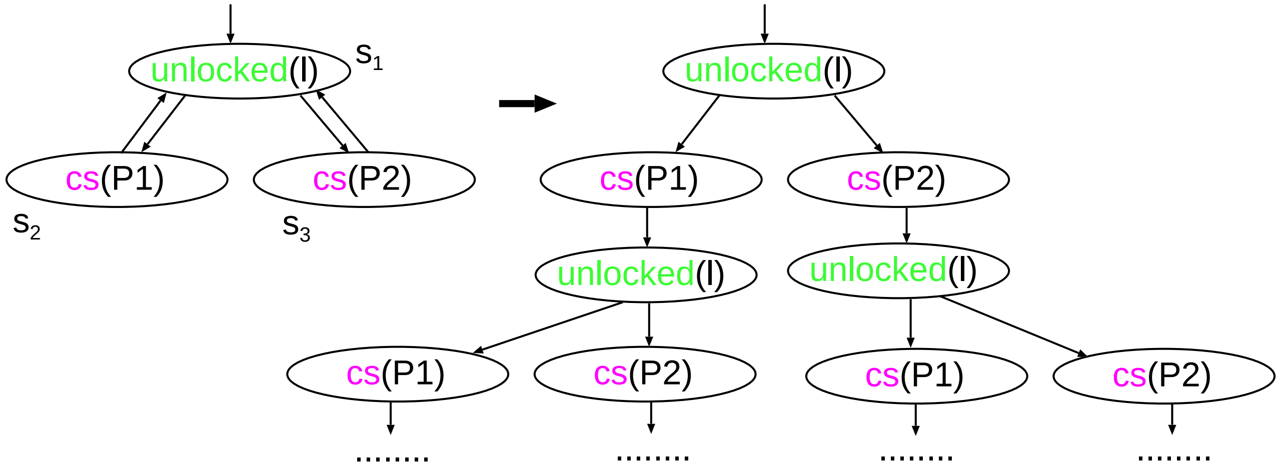
- The suffix π^i of a path $\pi = s_0 s_1 s_2 ... s_i s_{i+1} s_{i+2} ...$ is the path $\pi^i = s_i s_{i+1} s_{i+2} ...$ starting at s_i.

The CTL* Logic

CTL*—Basic Idea

❖ CTL* formulae describe properties of computation trees.

❖ Infinite computation trees are obtained by unwinding a Kripke structure from its initial states.



❖ CTL* formulae consist of:

- atomic propositions,
- Boolean connectives,
- path quantifiers,
- temporal operators.

CTL*—Quantifiers and Operators

- Path quantifiers—describe the branching structure of a computation tree:
  - E: for some computation path leading from a state,
  - A: for all computation paths leading from a state.

- Temporal operators—properties of a path through a computation tree:
  - $X\ \varphi$ ("next time", $\bigcirc$): the property φ holds (on the path starting) from the second state of the given path,
  - $F\ \varphi$ ("eventually" / "sometimes", $\Diamond$): the property φ holds (on the path starting) from some state of the given path,
  - $G\ \varphi$ ("always" / "globally", $\Box$): the property φ holds from all states of the path,
  - $\varphi\ U\ \psi$ ("until"): the property ψ holds from some state of the path, and the property φ holds from all preceding states of the path,
  - $\varphi\ R\ \psi$ ("release"): the property ψ holds from all states of the path up to (and including) the first state from where the property φ holds (if such a state exists).

CTL^*—The Syntax

❖ Let AP be a non-empty set of atomic propositions.

❖ The syntax of state formulae, which are true in a specific state, is given by the following rules:

- If $p \in AP$, then p is a state formula.
- If φ and ψ are state formulae, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$ are state formulae.
- If φ is a path formula, then $E\ \varphi$ and $A\ \varphi$ are state formulae.

❖ The syntax of path formulae, which are true along a specific path, is given by the following rules:

- If φ is a state formula, then φ is a path formula too.
- If φ and ψ are path formulae, then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $X\ \varphi$, $F\ \varphi$, $G\ \varphi$, $\varphi U \psi$, and $\varphi R \psi$ are path formulae.

❖ CTL* is the set of state formulae generated by the above rules.

CTL*—The Semantics

- Let a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP be given.
- For a *state formula* φ over AP, we denote $M, s \models \varphi$ the fact that φ holds at $s \in S$.
- For a *path formula* φ over AP, we denote $M, \pi \models \varphi$ the fact that φ holds along a path π in M.

CTL*—The Semantics

❖ Let a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP be given.

❖ For a *state formula* φ over AP, we denote $M, s \models \varphi$ the fact that φ holds at $s \in S$.

❖ For a *path formula* φ over AP, we denote $M, \pi \models \varphi$ the fact that φ holds along a path π in M.

❖ Let $s \in S$, π be a path in M, φ_1, φ_2 be state formulae over AP, $p \in AP$, and ψ_1, ψ_2 be path formulae over AP. We define the relation $\models$ inductively as follows:

Continued at the next slide...

CTL^*—The Semantics

❖ Let a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP be given.

❖ For a *state formula* φ over AP, we denote $M, s \models \varphi$ the fact that φ holds at $s \in S$.

❖ For a *path formula* φ over AP, we denote $M, \pi \models \varphi$ the fact that φ holds along a path π in M.

❖ Let $s \in S$, π be a path in M, φ_1, φ_2 be state formulae over AP, $p \in AP$, and ψ_1, ψ_2 be path formulae over AP. We define the relation $\models$ inductively as follows:

- $M, s \models p$ iff $p \in L(s)$.

Continued at the next slide...

CTL^*—The Semantics

❖ Let a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP be given.

❖ For a *state formula* φ over AP, we denote $M, s \models \varphi$ the fact that φ holds at $s \in S$.

❖ For a *path formula* φ over AP, we denote $M, \pi \models \varphi$ the fact that φ holds along a path π in M.

❖ Let $s \in S$, π be a path in M, φ_1, φ_2 be state formulae over AP, $p \in AP$, and ψ_1, ψ_2 be path formulae over AP. We define the relation $\models$ inductively as follows:

- $M, s \models p$ iff $p \in L(s)$.
- $M, s \models \neg\varphi_1$ iff $M, s \not\models \varphi_1$.

Continued at the next slide...

CTL^*—The Semantics

❖ Let a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP be given.

❖ For a *state formula* φ over AP, we denote $M, s \models \varphi$ the fact that φ holds at $s \in S$.

❖ For a *path formula* φ over AP, we denote $M, \pi \models \varphi$ the fact that φ holds along a path π in M.

❖ Let $s \in S$, π be a path in M, φ_1, φ_2 be state formulae over AP, $p \in AP$, and ψ_1, ψ_2 be path formulae over AP. We define the relation $\models$ inductively as follows:

- $M, s \models p$ iff $p \in L(s)$.
- $M, s \models \neg\varphi_1$ iff $M, s \not\models \varphi_1$.
- $M, s \models \varphi_1 \vee \varphi_2$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.

Continued at the next slide...

CTL*—The Semantics

❖ Let a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP be given.

❖ For a *state formula* φ over AP, we denote $M, s \models \varphi$ the fact that φ holds at $s \in S$.

❖ For a *path formula* φ over AP, we denote $M, \pi \models \varphi$ the fact that φ holds along a path π in M.

❖ Let $s \in S$, π be a path in M, φ_1, φ_2 be state formulae over AP, $p \in AP$, and ψ_1, ψ_2 be path formulae over AP. We define the relation $\models$ inductively as follows:

- $M, s \models p$ iff $p \in L(s)$.
- $M, s \models \neg\varphi_1$ iff $M, s \not\models \varphi_1$.
- $M, s \models \varphi_1 \vee \varphi_2$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
- $M, s \models \varphi_1 \wedge \varphi_2$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.

Continued at the next slide...

CTL*—The Semantics

- Let a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP be given.

- For a *state formula* φ over AP, we denote $M, s \models \varphi$ the fact that φ holds at $s \in S$.

- For a *path formula* φ over AP, we denote $M, \pi \models \varphi$ the fact that φ holds along a path π in M.

- Let $s \in S$, π be a path in M, φ_1, φ_2 be state formulae over AP, $p \in AP$, and ψ_1, ψ_2 be path formulae over AP. We define the relation $\models$ inductively as follows:
  - $M, s \models p$ iff $p \in L(s)$.
  - $M, s \models \neg\varphi_1$ iff $M, s \not\models \varphi_1$.
  - $M, s \models \varphi_1 \vee \varphi_2$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $M, s \models \varphi_1 \wedge \varphi_2$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $M, s \models E\ \psi_1$ iff $\exists \pi \in \Pi(M, s).\ M, \pi \models \psi_1$.

Continued at the next slide...

CTL*—The Semantics

- Let a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP be given.
- For a *state formula* φ over AP, we denote $M, s \models \varphi$ the fact that φ holds at $s \in S$.
- For a *path formula* φ over AP, we denote $M, \pi \models \varphi$ the fact that φ holds along a path π in M.
- Let $s \in S$, π be a path in M, φ_1, φ_2 be state formulae over AP, $p \in AP$, and ψ_1, ψ_2 be path formulae over AP. We define the relation $\models$ inductively as follows:
  - $M, s \models p$ iff $p \in L(s)$.
  - $M, s \models \neg\varphi_1$ iff $M, s \not\models \varphi_1$.
  - $M, s \models \varphi_1 \vee \varphi_2$ iff $M, s \models \varphi_1$ or $M, s \models \varphi_2$.
  - $M, s \models \varphi_1 \wedge \varphi_2$ iff $M, s \models \varphi_1$ and $M, s \models \varphi_2$.
  - $M, s \models E\ \psi_1$ iff $\exists \pi \in \Pi(M, s).\ M, \pi \models \psi_1$.
  - $M, s \models A\ \psi_1$ iff $\forall \pi \in \Pi(M, s).\ M, \pi \models \psi_1$.

Continued at the next slide...

CTL*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.

CTL*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.

CTL*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
- $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.

CTL*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
- $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
- $M, \pi \models \psi_1 \wedge \psi_2$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.

CTL*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
- $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
- $M, \pi \models \psi_1 \wedge \psi_2$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
- $M, \pi \models X\ \psi_1$ iff $M, \pi^1 \models \psi_1$.

CTL*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
- $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
- $M, \pi \models \psi_1 \wedge \psi_2$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
- $M, \pi \models X\ \psi_1$ iff $M, \pi^1 \models \psi_1$.
- $M, \pi \models F\ \psi_1$ iff $\exists i \geq 0.\ M, \pi^i \models \psi_1$.

CTL*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
- $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
- $M, \pi \models \psi_1 \wedge \psi_2$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
- $M, \pi \models X\ \psi_1$ iff $M, \pi^1 \models \psi_1$.
- $M, \pi \models F\ \psi_1$ iff $\exists i \geq 0.\ M, \pi^i \models \psi_1$.
- $M, \pi \models G\ \psi_1$ iff $\forall i \geq 0.\ M, \pi^i \models \psi_1$.

CTL*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
- $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
- $M, \pi \models \psi_1 \wedge \psi_2$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
- $M, \pi \models X\ \psi_1$ iff $M, \pi^1 \models \psi_1$.
- $M, \pi \models F\ \psi_1$ iff $\exists i \geq 0.\ M, \pi^i \models \psi_1$.
- $M, \pi \models G\ \psi_1$ iff $\forall i \geq 0.\ M, \pi^i \models \psi_1$.
- $M, \pi \models \psi_1\ U\ \psi_2$ iff $\exists i \geq 0.\ M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i.\ M, \pi^j \models \psi_1$.

CTL^*—The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
- $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
- $M, \pi \models \psi_1 \wedge \psi_2$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
- $M, \pi \models X\ \psi_1$ iff $M, \pi^1 \models \psi_1$.
- $M, \pi \models F\ \psi_1$ iff $\exists i \geq 0.\ M, \pi^i \models \psi_1$.
- $M, \pi \models G\ \psi_1$ iff $\forall i \geq 0.\ M, \pi^i \models \psi_1$.
- $M, \pi \models \psi_1\ U\ \psi_2$ iff $\exists i \geq 0.\ M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i.\ M, \pi^j \models \psi_1$.
- $M, \pi \models \psi_1\ R\ \psi_2$ iff $\forall i \geq 0.\ (\forall 0 \leq j < i.\ M, \pi^j \not\models \psi_1) \Rightarrow M, \pi^i \models \psi_2$.

CTL* —The Semantics

Continued from the previous slide...

- $M, \pi \models \varphi_1$ iff $M, s_0 \models \varphi_1$ where s_0 is the first state of π.
- $M, \pi \models \neg\psi_1$ iff $M, \pi \not\models \psi_1$.
- $M, \pi \models \psi_1 \vee \psi_2$ iff $M, \pi \models \psi_1$ or $M, \pi \models \psi_2$.
- $M, \pi \models \psi_1 \wedge \psi_2$ iff $M, \pi \models \psi_1$ and $M, \pi \models \psi_2$.
- $M, \pi \models X\ \psi_1$ iff $M, \pi^1 \models \psi_1$.
- $M, \pi \models F\ \psi_1$ iff $\exists i \geq 0.\ M, \pi^i \models \psi_1$.
- $M, \pi \models G\ \psi_1$ iff $\forall i \geq 0.\ M, \pi^i \models \psi_1$.
- $M, \pi \models \psi_1\ U\ \psi_2$ iff $\exists i \geq 0.\ M, \pi^i \models \psi_2$ and $\forall 0 \leq j < i.\ M, \pi^j \models \psi_1$.
- $M, \pi \models \psi_1\ R\ \psi_2$ iff $\forall i \geq 0.\ (\forall 0 \leq j < i.\ M, \pi^j \not\models \psi_1) \Rightarrow M, \pi^i \models \psi_2$.

❖ For a (state) CTL* formula φ, we write $M \models \varphi$ iff $\forall s_0 \in S_0.\ M, s_0 \models \varphi$.

CTL*—Basic Operators

❖ Provided that $AP \neq \emptyset$, it is easy to see that all CTL* operators can be derived from $\vee, \neg, X, U$, and E:

- let $p \in AP$, $true \equiv$ (and $false \equiv \neg true$),
- $\varphi \wedge \psi \equiv$
- $F\,\varphi \equiv$
- $G\,\varphi \equiv$
- $\varphi\, R\, \psi \equiv$
- $A\,\varphi \equiv$

CTL*—Basic Operators

❖ Provided that $AP \neq \emptyset$, it is easy to see that all CTL* operators can be derived from $\vee, \neg, X, U$, and E:

- let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$),
- $\varphi \wedge \psi \equiv$
- $F\, \varphi \equiv$
- $G\, \varphi \equiv$
- $\varphi\, R\, \psi \equiv$
- $A\, \varphi \equiv$

CTL*—Basic Operators

❖ Provided that $AP \neq \emptyset$, it is easy to see that all CTL* operators can be derived from $\vee, \neg, X, U$, and E:

- let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$),
- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$
- $F\ \varphi \equiv$
- $G\ \varphi \equiv$
- $\varphi\ R\ \psi \equiv$
- $A\ \varphi \equiv$

CTL^*—Basic Operators

❖ Provided that $AP \neq \emptyset$, it is easy to see that all CTL^* operators can be derived from $\vee, \neg, X, U$, and E:

- let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$),
- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$
- $F\ \varphi \equiv true\ U\ \varphi$,
- $G\ \varphi \equiv$
- $\varphi\ R\ \psi \equiv$
- $A\ \varphi \equiv$

CTL*—Basic Operators

❖ Provided that $AP \neq \emptyset$, it is easy to see that all CTL* operators can be derived from $\vee, \neg, X, U$, and E:

- let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$),
- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$
- $F\ \varphi \equiv true\ U\ \varphi$,
- $G\ \varphi \equiv \neg F\ \neg\varphi$,
- $\varphi\ R\ \psi \equiv$
- $A\ \varphi \equiv$

CTL^*—Basic Operators

❖ Provided that $AP \neq \emptyset$, it is easy to see that all CTL^* operators can be derived from $\vee, \neg, X, U$, and E:

- let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$),
- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$
- $F\ \varphi \equiv true\ U\ \varphi$,
- $G\ \varphi \equiv \neg F\ \neg\varphi$,
- $\varphi\ R\ \psi \equiv \neg(\neg\varphi\ U\ \neg\psi)$,
- $A\ \varphi \equiv$

CTL*—Basic Operators

❖ Provided that $AP \neq \emptyset$, it is easy to see that all CTL* operators can be derived from $\vee, \neg, X, U$, and E:

- let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$),
- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$
- $F\ \varphi \equiv true\ U\ \varphi$,
- $G\ \varphi \equiv \neg F\ \neg\varphi$,
- $\varphi\ R\ \psi \equiv \neg(\neg\varphi\ U\ \neg\psi)$,
- $A\ \varphi \equiv \neg E\ \neg\varphi$.

CTL*—Basic Operators

❖ Provided that $AP \neq \emptyset$, it is easy to see that all CTL* operators can be derived from $\vee, \neg, X, U$, and E:

- let $p \in AP$, $true \equiv p \vee \neg p$ (and $false \equiv \neg true$),
- $\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi)$
- $F\ \varphi \equiv true\ U\ \varphi$,
- $G\ \varphi \equiv \neg F\ \neg\varphi$,
- $\varphi\ R\ \psi \equiv \neg(\neg\varphi\ U\ \neg\psi)$,
- $A\ \varphi \equiv \neg E\ \neg\varphi$.

❖ Some further connectives may be introduced too, e.g.:

- $\varphi \Rightarrow \psi \equiv \neg\varphi \vee \psi$,
- $\varphi \Leftrightarrow \psi \equiv (\varphi \Rightarrow \psi) \wedge (\psi \Rightarrow \varphi)$,
- ...

The CTL Logic

CTL—The Syntax

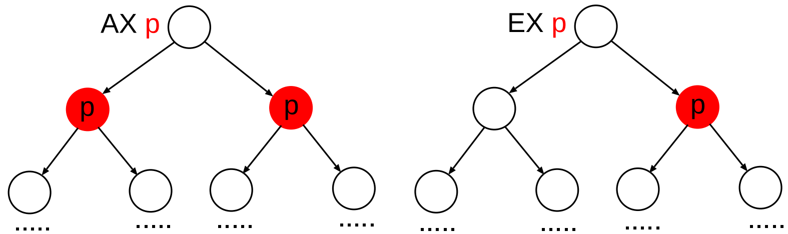
❖ CTL is a sublogic of CTL*—the path formulae are restricted to $X\ \varphi$, $F\ \varphi$, $G\ \varphi$, $\varphi U \psi$, and $\varphi R \psi$ for φ, ψ being state formulae.

CTL—The Syntax

❖ CTL is a sublogic of CTL*—the path formulae are restricted to $X\ \varphi$, $F\ \varphi$, $G\ \varphi$, $\varphi U \psi$, and $\varphi R \psi$ for φ, ψ being state formulae.

❖ In effect, there are allowed these 10 modal CTL operators:

- AX and EX,



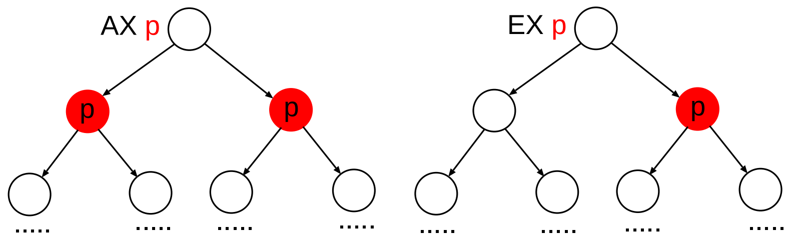
Continued at the next slide...

CTL—The Syntax

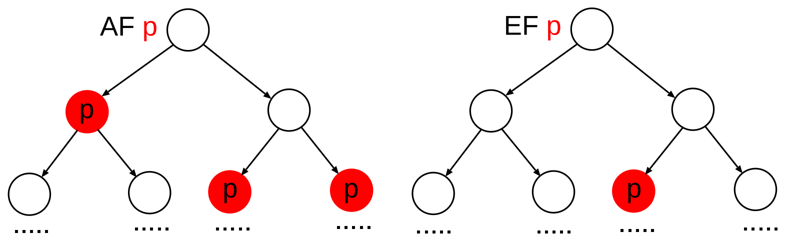
❖ CTL is a sublogic of CTL*—the path formulae are restricted to $X\ \varphi$, $F\ \varphi$, $G\ \varphi$, $\varphi U \psi$, and $\varphi R \psi$ for φ, ψ being state formulae.

❖ In effect, there are allowed these 10 modal CTL operators:

- AX and EX,



- AF and EF,



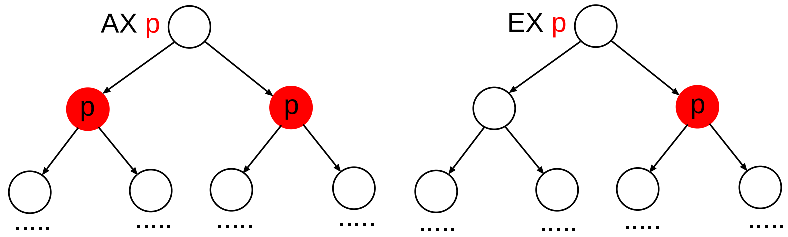
Continued at the next slide...

CTL—The Syntax

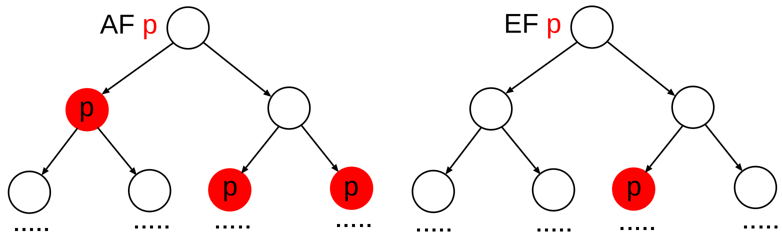
❖ CTL is a sublogic of CTL*—the path formulae are restricted to $X\ \varphi$, $F\ \varphi$, $G\ \varphi$, $\varphi U \psi$, and $\varphi R \psi$ for φ, ψ being state formulae.

❖ In effect, there are allowed these 10 modal CTL operators:

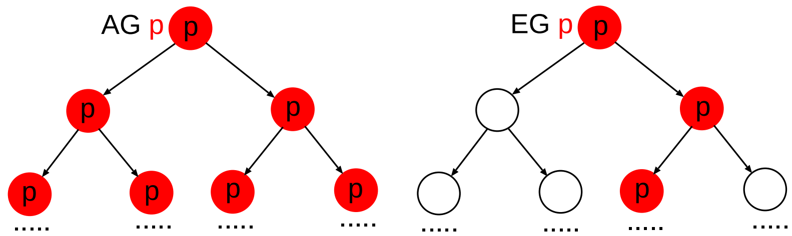
- AX and EX,



- AF and EF,



- AG and EG,

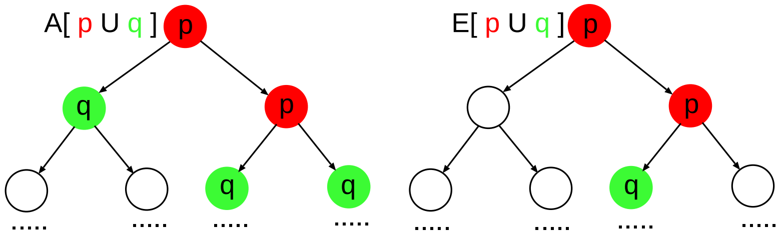


Continued at the next slide...

CTL—The Syntax

Continued from the previous slide...

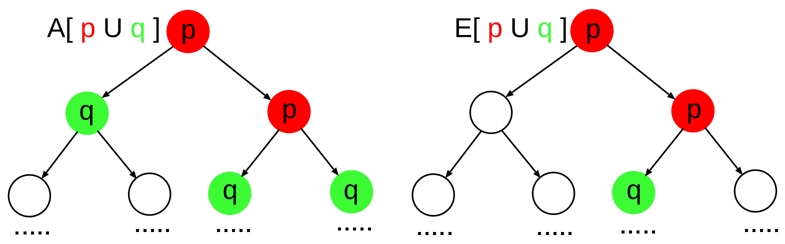
- AU and EU,



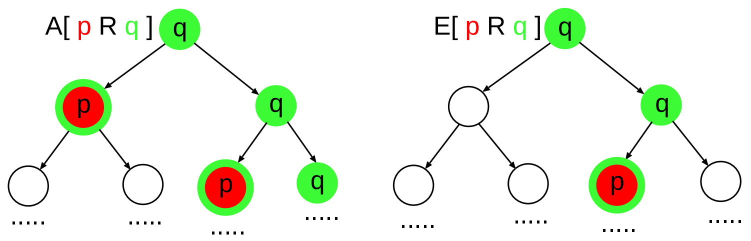
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



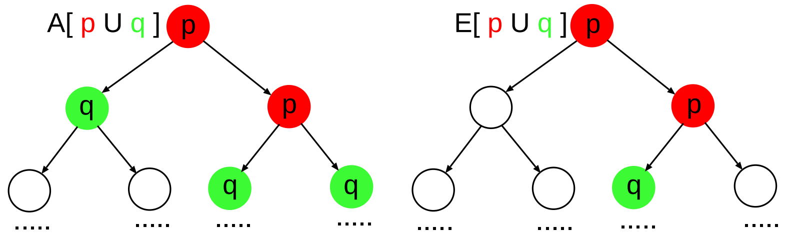
- AR and ER.



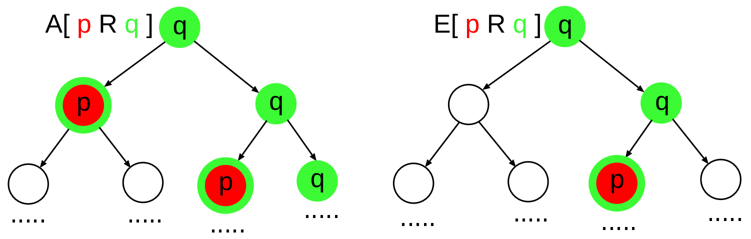
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



- AR and ER.



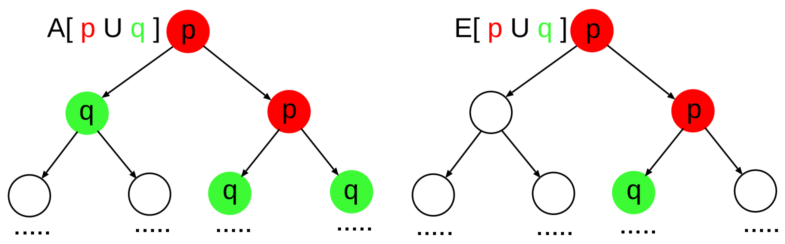
❖ There are 3 basic CTL modal operators—EX, EG, and EU:

- $AX\ \varphi \equiv$
- $EF\ \varphi \equiv$
- $AG\ \varphi \equiv$
- $AF\ \varphi \equiv$
- $A[\varphi\ U\ \psi] \equiv$
- $A[\varphi\ R\ \psi] \equiv$
- $E[\varphi\ R\ \psi] \equiv$

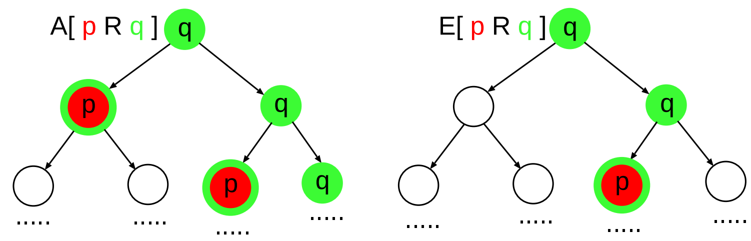
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



- AR and ER.



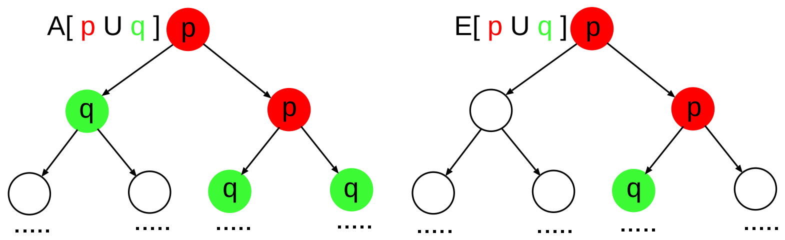
❖ There are 3 basic CTL modal operators—EX, EG, and EU:

- $AX\ \varphi \equiv \neg EX\ \neg\varphi$,
- $EF\ \varphi \equiv$
- $AG\ \varphi \equiv$
- $AF\ \varphi \equiv$
- $A[\varphi\ U\ \psi] \equiv$
- $A[\varphi\ R\ \psi] \equiv$
- $E[\varphi\ R\ \psi] \equiv$

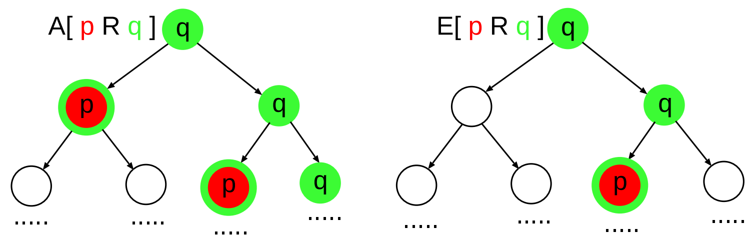
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



- AR and ER.



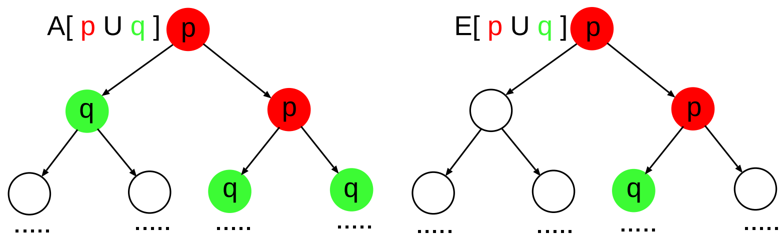
❖ There are 3 basic CTL modal operators—EX, EG, and EU:

- $AX\ \varphi \equiv \neg EX\ \neg\varphi$,
- $EF\ \varphi \equiv E[true\ U\ \varphi]$,
- $AG\ \varphi \equiv$
- $AF\ \varphi \equiv$
- $A[\varphi\ U\ \psi] \equiv$
- $A[\varphi\ R\ \psi] \equiv$
- $E[\varphi\ R\ \psi] \equiv$

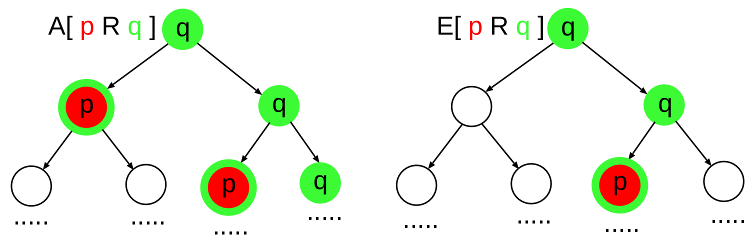
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



- AR and ER.



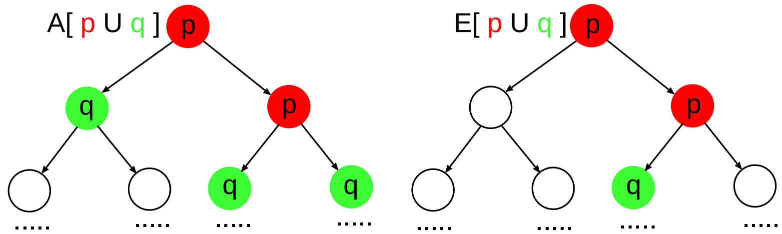
❖ There are 3 basic CTL modal operators—EX, EG, and EU:

- $AX\ \varphi \equiv \neg EX\ \neg\varphi$,
- $EF\ \varphi \equiv E[true\ U\ \varphi]$,
- $AG\ \varphi \equiv \neg EF\ \neg\varphi$,
- $AF\ \varphi \equiv$
- $A[\varphi\ U\ \psi] \equiv$
- $A[\varphi\ R\ \psi] \equiv$
- $E[\varphi\ R\ \psi] \equiv$

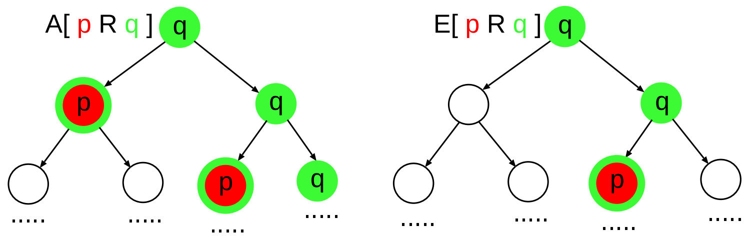
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



- AR and ER.



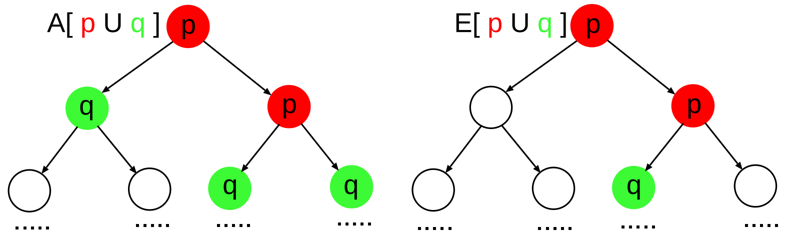
❖ There are 3 basic CTL modal operators—EX, EG, and EU:

- $AX\ \varphi \equiv \neg EX\ \neg\varphi$,
- $EF\ \varphi \equiv E[true\ U\ \varphi]$,
- $AG\ \varphi \equiv \neg EF\ \neg\varphi$,
- $AF\ \varphi \equiv \neg EG\ \neg\varphi$,
- $A[\varphi\ U\ \psi] \equiv$
- $A[\varphi\ R\ \psi] \equiv$
- $E[\varphi\ R\ \psi] \equiv$

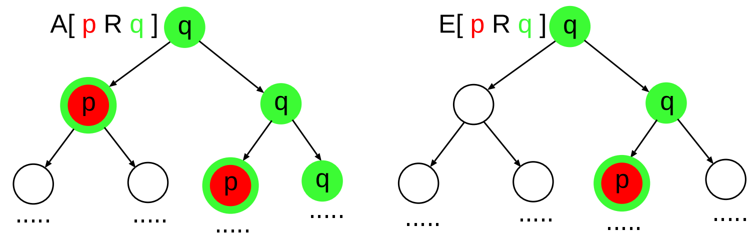
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



- AR and ER.



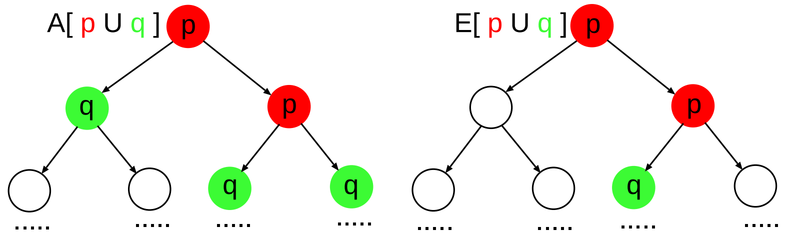
❖ There are 3 basic CTL modal operators—EX, EG, and EU:

- $AX\ \varphi \equiv \neg EX\ \neg\varphi$,
- $EF\ \varphi \equiv E[true\ U\ \varphi]$,
- $AG\ \varphi \equiv \neg EF\ \neg\varphi$,
- $AF\ \varphi \equiv \neg EG\ \neg\varphi$,
- $A[\varphi\ U\ \psi] \equiv \neg E[\neg\psi\ U\ (\neg\varphi \wedge \neg\psi)]\ \wedge\ AF\psi$,
- $A[\varphi\ R\ \psi] \equiv$
- $E[\varphi\ R\ \psi] \equiv$

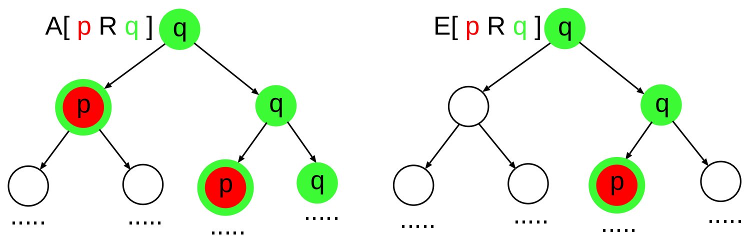
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



- AR and ER.



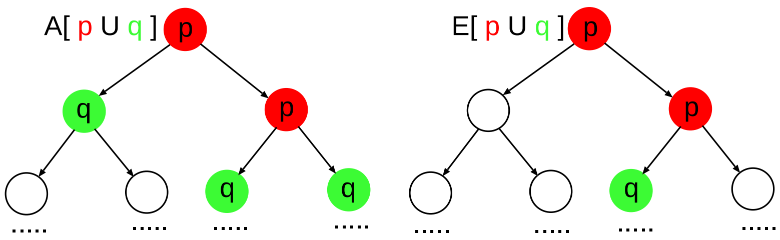
❖ There are 3 basic CTL modal operators—EX, EG, and EU:

- $AX\ \varphi \equiv \neg EX\ \neg\varphi$,
- $EF\ \varphi \equiv E[true\ U\ \varphi]$,
- $AG\ \varphi \equiv \neg EF\ \neg\varphi$,
- $AF\ \varphi \equiv \neg EG\ \neg\varphi$,
- $A[\varphi\ U\ \psi] \equiv \neg E[\neg\psi\ U\ (\neg\varphi \wedge \neg\psi)]\ \wedge\ AF\psi$,
- $A[\varphi\ R\ \psi] \equiv \neg E[\neg\varphi\ U\ \neg\psi]$,
- $E[\varphi\ R\ \psi] \equiv$

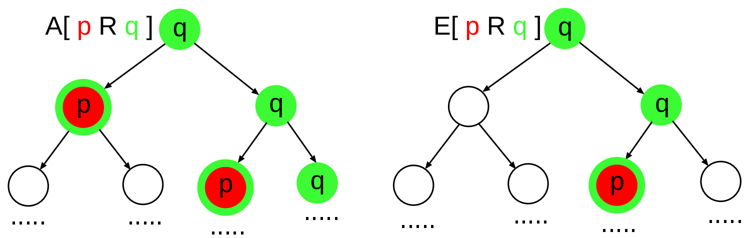
CTL—The Syntax

Continued from the previous slide...

- AU and EU,



- AR and ER.



❖ There are 3 basic CTL modal operators—EX, EG, and EU:

- $AX\ \varphi \equiv \neg EX\ \neg\varphi$,
- $EF\ \varphi \equiv E[true\ U\ \varphi]$,
- $AG\ \varphi \equiv \neg EF\ \neg\varphi$,
- $AF\ \varphi \equiv \neg EG\ \neg\varphi$,
- $A[\varphi\ U\ \psi] \equiv \neg E[\neg\psi\ U\ (\neg\varphi \wedge \neg\psi)]\ \wedge\ AF\psi$,
- $A[\varphi\ R\ \psi] \equiv \neg E[\neg\varphi\ U\ \neg\psi]$,
- $E[\varphi\ R\ \psi] \equiv \neg A[\neg\varphi\ U\ \neg\psi]$.

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.

CTL—Some Examples

❖ Some examples of CTL formulae:

- Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
  $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.
    $EF\ (Start\ \wedge\ \neg Ready)$

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.
    $EF\ (Start\ \wedge\ \neg Ready)$
  - Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).

CTL—Some Examples

❖ Some examples of CTL formulae:

- Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
  $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
- It is possible to get to a state where $Start$ holds, but $Ready$ does not.
  $EF\ (Start\ \wedge\ \neg Ready)$
- Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
  $AG\ (Req\ \Rightarrow\ AF\ Ack)$

CTL—Some Examples

❖ Some examples of CTL formulae:

- Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
  $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
- It is possible to get to a state where $Start$ holds, but $Ready$ does not.
  $EF\ (Start\ \wedge\ \neg Ready)$
- Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
  $AG\ (Req\ \Rightarrow\ AF\ Ack)$
- In any run of the system, $DeviceEnabled$ is true infinitely often.

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.
    $EF\ (Start\ \wedge\ \neg Ready)$
  - Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
    $AG\ (Req\ \Rightarrow\ AF\ Ack)$
  - In any run of the system, $DeviceEnabled$ is true infinitely often.
    $AG\ AF\ DeviceEnabled$

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.
    $EF\ (Start\ \wedge\ \neg Ready)$
  - Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
    $AG\ (Req\ \Rightarrow\ AF\ Ack)$
  - In any run of the system, $DeviceEnabled$ is true infinitely often.
    $AG\ AF\ DeviceEnabled$
  - From any state, the system can be restarted (i.e. get to a $Restart$ state).

CTL—Some Examples

❖ Some examples of CTL formulae:

- Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
  $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
- It is possible to get to a state where $Start$ holds, but $Ready$ does not.
  $EF\ (Start\ \wedge\ \neg Ready)$
- Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
  $AG\ (Req\ \Rightarrow\ AF\ Ack)$
- In any run of the system, $DeviceEnabled$ is true infinitely often.
  $AG\ AF\ DeviceEnabled$
- From any state, the system can be restarted (i.e. get to a $Restart$ state).
  $AG\ EF\ Restart$

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.
    $EF\ (Start\ \wedge\ \neg Ready)$
  - Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
    $AG\ (Req\ \Rightarrow\ AF\ Ack)$
  - In any run of the system, $DeviceEnabled$ is true infinitely often.
    $AG\ AF\ DeviceEnabled$
  - From any state, the system can be restarted (i.e. get to a $Restart$ state).
    $AG\ EF\ Restart$
  - The $Reset$ signal is initially set, and from the next state on, it is never set again.

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.
    $EF\ (Start\ \wedge\ \neg Ready)$
  - Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
    $AG\ (Req\ \Rightarrow\ AF\ Ack)$
  - In any run of the system, $DeviceEnabled$ is true infinitely often.
    $AG\ AF\ DeviceEnabled$
  - From any state, the system can be restarted (i.e. get to a $Restart$ state).
    $AG\ EF\ Restart$
  - The $Reset$ signal is initially set, and from the next state on, it is never set again.
    $Reset\ \wedge\ AX\ AG\ \neg Reset$

CTL—Some Examples

❖ Some examples of CTL formulae:

- Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
  $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
- It is possible to get to a state where $Start$ holds, but $Ready$ does not.
  $EF\ (Start\ \wedge\ \neg Ready)$
- Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
  $AG\ (Req\ \Rightarrow\ AF\ Ack)$
- In any run of the system, $DeviceEnabled$ is true infinitely often.
  $AG\ AF\ DeviceEnabled$
- From any state, the system can be restarted (i.e. get to a $Restart$ state).
  $AG\ EF\ Restart$
- The $Reset$ signal is initially set, and from the next state on, it is never set again.
  $Reset\ \wedge\ AX\ AG\ \neg Reset$
- The $Reset$ signal is initially set, but once it is unset, it is never set again.

CTL—Some Examples

❖ Some examples of CTL formulae:

- Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
  $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
- It is possible to get to a state where $Start$ holds, but $Ready$ does not.
  $EF\ (Start\ \wedge\ \neg Ready)$
- Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
  $AG\ (Req\ \Rightarrow\ AF\ Ack)$
- In any run of the system, $DeviceEnabled$ is true infinitely often.
  $AG\ AF\ DeviceEnabled$
- From any state, the system can be restarted (i.e. get to a $Restart$ state).
  $AG\ EF\ Restart$
- The $Reset$ signal is initially set, and from the next state on, it is never set again.
  $Reset\ \wedge\ AX\ AG\ \neg Reset$
- The $Reset$ signal is initially set, but once it is unset, it is never set again.
  $Reset\ \wedge\ AG\ (\neg Reset\ \Rightarrow\ AG\ \neg Reset)$

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.
    $EF\ (Start\ \wedge\ \neg Ready)$
  - Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
    $AG\ (Req\ \Rightarrow\ AF\ Ack)$
  - In any run of the system, $DeviceEnabled$ is true infinitely often.
    $AG\ AF\ DeviceEnabled$
  - From any state, the system can be restarted (i.e. get to a $Restart$ state).
    $AG\ EF\ Restart$
  - The $Reset$ signal is initially set, and from the next state on, it is never set again.
    $Reset\ \wedge\ AX\ AG\ \neg Reset$
  - The $Reset$ signal is initially set, but once it is unset, it is never set again.
    $Reset\ \wedge\ AG\ (\neg Reset\ \Rightarrow\ AG\ \neg Reset)$
  - The $AccConn$ signal can be set only after the $StartAcc$ signal arrives.

CTL—Some Examples

- Some examples of CTL formulae:
  - Mutual exclusion of two processes using propositions $cs(P1)$ (process $P1$ is in the critical section) and $cs(P2)$.
    $\neg EF\ (cs(P1)\ \wedge\ cs(P2))\ \equiv\ AG\ (\neg cs(P1)\ \vee\ \neg cs(P2))$
  - It is possible to get to a state where $Start$ holds, but $Ready$ does not.
    $EF\ (Start\ \wedge\ \neg Ready)$
  - Whenever a request occurs (i.e. Req holds), then it will eventually be acknowledged (i.e. Ack will hold).
    $AG\ (Req\ \Rightarrow\ AF\ Ack)$
  - In any run of the system, $DeviceEnabled$ is true infinitely often.
    $AG\ AF\ DeviceEnabled$
  - From any state, the system can be restarted (i.e. get to a $Restart$ state).
    $AG\ EF\ Restart$
  - The $Reset$ signal is initially set, and from the next state on, it is never set again.
    $Reset\ \wedge\ AX\ AG\ \neg Reset$
  - The $Reset$ signal is initially set, but once it is unset, it is never set again.
    $Reset\ \wedge\ AG\ (\neg Reset\ \Rightarrow\ AG\ \neg Reset)$
  - The $AccConn$ signal can be set only after the $StartAcc$ signal arrives.
    $A[StartAcc\ R\ (\neg AccConn)]$

CTL Model Checking

The Basic Idea

❖ The CTL model checking question to be answered: Given a Kripke structure $M = (S, S_0, R, L)$ over a set of atomic propositions AP and a CTL formula φ over AP, does $M \models \varphi$ hold?

❖ A very basic approach to answer the CTL model checking question by the so-called explicit-state model checking:

- For every subformula ψ of φ, label by ψ all those states s of M in which φ holds (i.e., $M, s \models \psi$).
- Perform the labelling from the inner-most subformulae (i.e. the most nested ones) going to the outer ones exploiting the already computed labels (with atomic propositions corresponding to the original labels of M).
- Check whether each state in S_0 gets labelled by φ.

❖ It is enough to consider the basic operators of CTL, i.e. the below syntax for $p \in AP$:

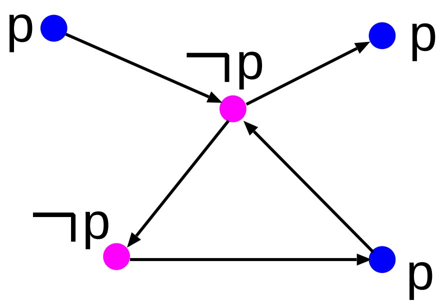
$$\varphi ::= p \mid \neg\varphi \mid \varphi \;\vee\; \varphi \mid EX\varphi \mid E[\varphi U \varphi] \mid EG\varphi.$$

Label(¬φ), Label(φ₁ ∨ φ₂)

Label($\neg\varphi$)

`for all` $s \in S$ such that $\varphi \notin Label(s)$ `do`

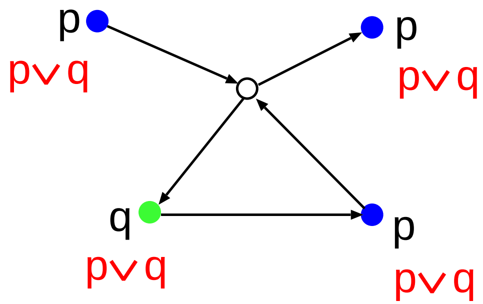
$Label(s) := Label(s) \cup \{\neg\varphi\}$



Label($\varphi_1 \vee \varphi_2$)

`for all` $s \in S$ such that $\varphi_1 \in Label(s)$ `or` $\varphi_2 \in Label(s)$ `do`

$Label(s) := Label(s) \cup \{\varphi_1 \vee \varphi_2\}$



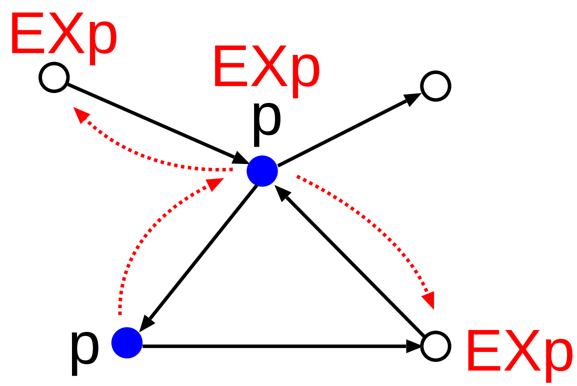
Label($EX\varphi$)

Label($EX\varphi$)

`for all` $s_2 \in S$ such that $\varphi \in Label(s_2)$ `do`

  `for all` $s_1 \in S$ such that $R(s_1, s_2)$ `do`

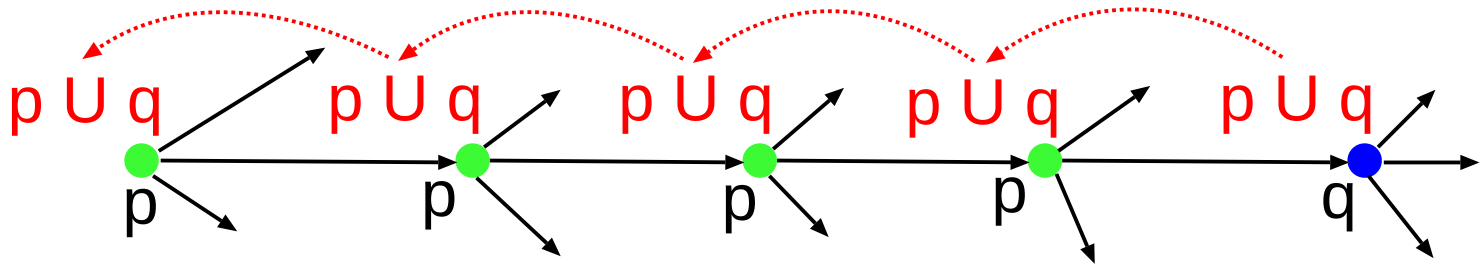
    $Label(s_1) := Label(s_1) \cup \{EX\varphi\}$



Label($E[\varphi_1 \ U \ \varphi_2]$)

❖ The idea:

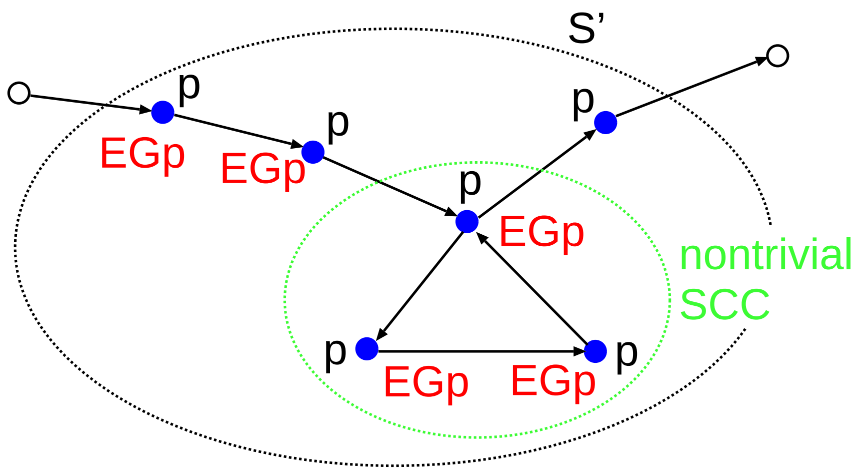
- Label first states already labelled by φ_2.
- Look at predecessors of states labelled with $\varphi_1 \ U \ \varphi_2$, and if they are labelled with φ_1, label them with $\varphi_1 \ U \ \varphi_2$ as well.



Label($EG\varphi$)

❖ Based on the following observation: Let $M = (S, S_0, R, L)$ be a Kripke structure, $S' = \{s \in S \mid M, s \models \varphi\}$, and $R' = R \cap (S' \times S')$. For any $s \in S$, $M, s \models EG\varphi$ iff

1. $s \in S'$ and
2. there exists a path in the oriented graph $G' = (S', R')$ that leads from s to some node t in a nontrivial SCC C of G'.



❖ An SCC C is nontrivial iff either it has more than one node or it contains one node with a self-loop.
❖ SCCs of a finite oriented graph (V, E) can be computed using the Tarjan's algorithm in time $O(|E| + |V|)$.

The LTL Logic

LTL—The Syntax

❖ LTL is another sublogic of CTL* that allows only formulae of the form $A\ \varphi$ in which the only state subformulae are atomic propositions.

❖ This is, LTL formulae φ are built according to the grammar:

- $\varphi ::= A\ \psi$ (the use of A is often omitted),
- $\psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid X\ \psi \mid F\ \psi \mid G\ \psi \mid \psi U \psi \mid \psi R \psi$

where $p \in AP$.

❖ Note that LTL speaks about particular paths in a given Kripke structure only—it ignores its branching structure.

❖ Sometimes, existential LTL allowing formulae of the form $E\ \varphi$ is used too.

LTL—The Syntax

❖ LTL is another sublogic of CTL* that allows only formulae of the form $A\ \varphi$ in which the only state subformulae are atomic propositions.

❖ This is, LTL formulae φ are built according to the grammar:

- $\varphi ::= A\ \psi$ (the use of A is often omitted),
- $\psi ::= p \mid \neg\psi \mid \psi \vee \psi \mid \psi \wedge \psi \mid X\ \psi \mid F\ \psi \mid G\ \psi \mid \psi U \psi \mid \psi R \psi$

where $p \in AP$.

❖ Note that LTL speaks about particular paths in a given Kripke structure only—it ignores its branching structure.

❖ Sometimes, existential LTL allowing formulae of the form $E\ \varphi$ is used too.

❖ Note also that while CTL cannot express fairness assumptions (in CTL model checking, they are handled by a special extension of the model checking algorithm), LTL can express fairness assumptions by formulae of the following form:

- weak fairness: $(F\ G\ Enabled) \Rightarrow\ (G\ F\ Fired)$, i.e. $\Diamond\Box\ Enabled \Rightarrow\ \Box\Diamond\ Fired$,
- strong fairness: $(G\ F\ Enabled) \Rightarrow\ (G\ F\ Fired)$, i.e. $\Box\Diamond\ Enabled \Rightarrow\ \Box\Diamond\ Fired$.

LTL, CTL, and CTL*

❖ LTL and CTL have an incomparable power:

- CTL cannot express, e.g., the LTL formula $A\,(FG\,p)$,
- LTL cannot express, e.g., the CTL formula $AG\,(EF\,p)$.

LTL, CTL, and CTL*

❖ LTL and CTL have an incomparable power:

- CTL cannot express, e.g., the LTL formula $A\ (FG\ p)$,
- LTL cannot express, e.g., the CTL formula $AG\ (EF\ p)$.

❖ CTL* is strictly more powerful than both LTL and CTL:

- the disjunction of the above formulae, i.e. $(A\ (FG\ p))\ \vee\ (AG\ (EF\ p))$, is not expressible in CTL nor LTL.

LTL, CTL, and CTL*

❖ LTL and CTL have an incomparable power:

- CTL cannot express, e.g., the LTL formula $A\ (FG\ p)$,
- LTL cannot express, e.g., the CTL formula $AG\ (EF\ p)$.

❖ CTL* is strictly more powerful than both LTL and CTL:

- the disjunction of the above formulae, i.e. $(A\ (FG\ p))\ \vee\ (AG\ (EF\ p))$, is not expressible in CTL nor LTL.

❖ To complete the picture, here are the complexities of the appropriate model checking algorithms (we will discuss LTL model checking later on):

- CTL: linear in $|M|$ and linear in $|\varphi|$.
- LTL and CTL*: linear in $|M|$ and PSPACE-complete in $|\varphi|$

where $|M| = |S| + |R|$ and $|\varphi|$ is the number of subformulae of φ.

❖ Finally, as an example of a logic more general than CTL*, we can mention modal μ-calculus based on least/greatest fixpoint operators on sets of states (basically allowing one to define new, specialised modalities).