Computation Tree Logic

Hao Zheng

Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620
Email: zheng@cse.usf.edu
Phone: (813)974-4757
Fax: (813)974-5456

Overview

Contents

Introduction (6.1)

- *Linear* temporal logic:

  "Statements about (all) paths starting in a state."

  - $s \models \Box(x \leq 20)$ iff for all possible paths starting in s always $x \leq 20$.
  - Quantifier $\forall$ is implicit: $s \models \Box(x \leq 20) \;\equiv\; s \models \forall\Box(x \leq 20)$

Introduction (6.1)

- *Linear* temporal logic:

  "Statements about (all) paths starting in a state."

  - $s \models \Box(x \leq 20)$ iff for all possible paths starting in s always $x \leq 20$.
  - Quantifier $\forall$ is implicit: $s \models \Box(x \leq 20) \ \equiv \ s \models \forall\Box(x \leq 20)$

- *Branching* temporal logic:

  "Statements about all or some paths starting in a state."

  - $s \models \forall\Box(x \leq 20)$ iff for **all** paths starting in s always $x \leq 20$.
  - $s \models \exists\Box(x \leq 20)$ iff for **some** path starting in s always $x \leq 20$.
  - Nesting of path quantifiers is allowed.

Introduction (6.1)

- *Linear* temporal logic:

  "Statements about (all) paths starting in a state."

  - $s \models \Box(x \leq 20)$ iff for all possible paths starting in s always $x \leq 20$.
  - Quantifier $\forall$ is implicit: $s \models \Box(x \leq 20) \;\equiv\; s \models \forall\Box(x \leq 20)$
- *Branching* temporal logic:

  "Statements about all or some paths starting in a state."

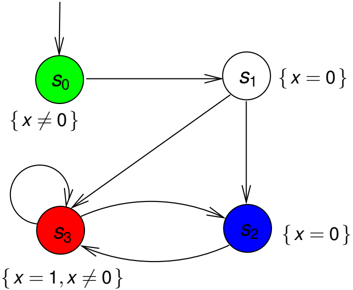
  - $s \models \forall\Box(x \leq 20)$ iff for **all** paths starting in s always $x \leq 20$.
  - $s \models \exists\Box(x \leq 20)$ iff for **some** path starting in s always $x \leq 20$.
  - Nesting of path quantifiers is allowed.
- Checking $\exists\varphi$ in LTL can be done using $\forall\neg\varphi$, but this does not work for nested formulas such as $\forall\Box\exists\Diamond a$.

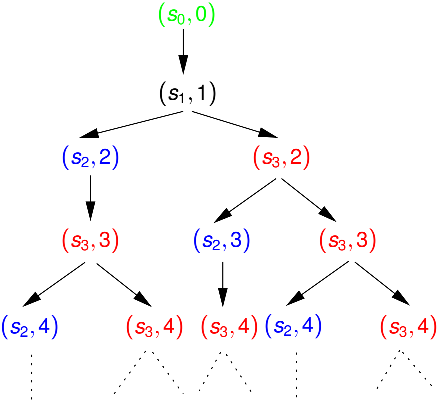
  In any state of every computation ($\forall\Box$), it is possible ($\exists\Diamond$) to return to the initial state.

$\Box\Diamond a$ vs $\forall\Box\exists\Diamond a$, differenece?

Computational Tree View of Transition Systems

- Semantics is based on a branching notion of time.
  - An infinite tree of states obtained by unfolding the transition system.
  - One "time instant" may have several possible successor "time instants".





Branching vs Linear Temporal Logics

- Incomparable expressiveness:
  - There are properties that can be expressed in LTL, but not in CTL.
  - There are also properties that can be expressed in CTL, but not in LTL.
- Distinct model-checking algorithms with different time/space complexities.
- Fairness assumptions require special treatment in CTL.
  - A natural part of LTL.
- Equivalences and preorders between transition systems based on simulation and bisimulation relations rather than traces.

Contents

Computational Tree Logic - Syntax (6.2.1)

Modal logic over infinite trees [Clarke & Emerson 1981].

- Statements over states (Φ):
  - $a \in AP$ — atomic proposition
  - $\neg \Phi$ and $\Phi_1 \wedge \Phi_2$ — negation and conjunction
  - $\exists\varphi$ — there *exists* a path fulfilling φ
  - $\forall\varphi$ — *all* paths fulfill φ
- Statements over paths (φ):
  - $\bigcirc\Phi$ — the next state fulfills Φ
  - $\Phi_1 \,\mathsf{U}\, \Phi_2$ — Φ_1 holds until a Φ_2-state is reached

Computational Tree Logic - Syntax (6.2.1)

Modal logic over infinite trees [Clarke & Emerson 1981].

- Statements over states (Φ):
  - $a \in AP$ — atomic proposition
  - $\neg \Phi$ and $\Phi_1 \wedge \Phi_2$ — negation and conjunction
  - $\exists \varphi$ — there *exists* a path fulfilling φ
  - $\forall \varphi$ — *all* paths fulfill φ
- Statements over paths (φ):
  - $\bigcirc \Phi$ — the next state fulfills Φ
  - $\Phi_1 \, \mathsf{U} \, \Phi_2$ — Φ_1 holds until a Φ_2-state is reached

$\Rightarrow$ Note that $\bigcirc$ and U *alternate* with $\forall$ and $\exists$:

- $\forall \bigcirc \bigcirc \Phi$, $\forall \exists \bigcirc \Phi \notin$ CTL, but $\forall \bigcirc \forall \bigcirc \Phi$ and $\forall \bigcirc \exists \bigcirc \Phi \in$ CTL.
- Four operators by the syntax rules:

$$\begin{array}{llll} \forall \bigcirc \ (\mathsf{AX}), & \forall \Box \ (\mathsf{AG}), & \forall \mathsf{U} \ (\mathsf{AU}), & \forall \Diamond \ (\mathsf{AF}) \\ \exists \bigcirc \ (\mathsf{EX}), & \exists \Box \ (\mathsf{EG}), & \exists \mathsf{U} \ (\mathsf{EU}), & \exists \Diamond \ (\mathsf{EF}) \end{array}$$

- Check Example 6.2 in the book for some example formulas.

Derived Operators

potentially Φ:	$\exists\Diamond\Phi$	$=$	$\exists(\mathsf{true}\,\mathsf{U}\,\Phi)$
inevitably Φ:	$\forall\Diamond\Phi$	$=$	$\forall(\mathsf{true}\,\mathsf{U}\,\Phi)$
potentially always Φ:	$\exists\Box\Phi$	$=$	$\neg\forall\Diamond\neg\Phi$
invariantly Φ:	$\forall\Box\Phi$	$=$	$\neg\exists\Diamond\neg\Phi$
weak until:	$\exists(\Phi_1\,\mathsf{U}\,\Phi_2)$	$=$	$\neg\forall\big((\Phi_1 \;\wedge\; \neg\Phi_2)\,\mathsf{U}\,(\neg\Phi_1 \;\wedge\; \neg\Phi_2)\big)$
	$\forall(\Phi_1\,\mathsf{U}\,\Phi_2)$	$=$	$\neg\exists\big((\Phi_1 \;\wedge\; \neg\Phi_2)\,\mathsf{U}\,(\neg\Phi_1 \;\wedge\; \neg\Phi_2)\big)$

The boolean connectives are derived as usual.

Example Properties in CTL

- Mutual exclusion:

$$\forall\Box(\neg crit_1 \vee \neg crit_2)$$

- Starvation freedom:

$$(\forall\Box\forall\Diamond crit_1) \;\wedge\; (\forall\Box\forall\Diamond crit_2)$$

- Each red light is preceded by a yellow light:

$$\forall\Box(yellow \;\vee\; \forall \bigcirc \neg red)???$$

- Traffic light is infinitely often green:

$$\forall\Box\forall\Diamond green$$

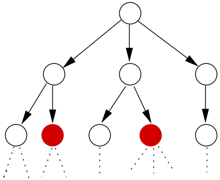
- Every request is eventually granted:

$$\forall\Box(request \;\Rightarrow\; \forall\Diamond response)$$

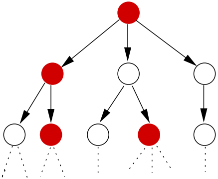
- In every reachable state, it is possible to return to the start state:

$$\forall\Box\exists\Diamond start$$

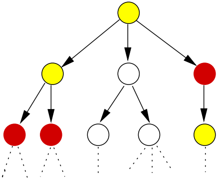
CTL Semantics Visualization



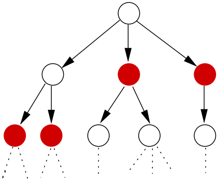
$\exists\Diamond\, \textit{red}$



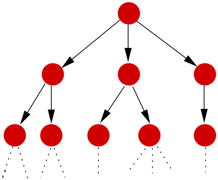
$\exists\Box\, \textit{red}$



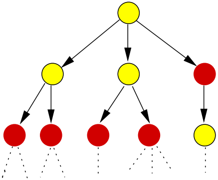
$\exists(\textit{yellow}\, \mathsf{U}\, \textit{red})$



$\forall\Diamond\, \textit{red}$



$\forall\Box\, \textit{red}$



$\forall(\textit{yellow}\, \mathsf{U}\, \textit{red})$

CTL Semantics - State Formulas

Defined by a relation $\models$ such that

$$\boxed{s \models \Phi \text{ if and only if formula } \Phi \text{ holds in state } s}$$

$s \models a$	iff	$a \in L(s)$
$s \models \neg\,\Phi$	iff	$\neg\,(s \models \Phi)$
$s \models \Phi \ \wedge \ \Psi$	iff	$(s \models \Phi) \ \wedge \ (s \models \Psi)$
$s \models \exists\varphi$	iff	$\pi \models \varphi$ for *some* path π that starts in s
$s \models \forall\varphi$	iff	$\pi \models \varphi$ for *all* paths π that start in s

CTL Semantics - Path Formulas

Define a relation $\models$ such that

$\boxed{\pi \models \varphi \text{ if and only if path } \pi \text{ satisfies } \varphi}$

$$\pi \models \bigcirc\Phi \qquad \text{iff } \pi[1] \models \Phi$$

$$\pi \models \Phi \,\mathsf{U}\, \Psi \qquad \text{iff } (\exists\, j \geq 0.\, \pi[j] \models \Psi \;\wedge\; (\forall\, 0 \leq k < j.\, \pi[k] \models \Phi))$$

where $\pi[i]$ denotes the state s_i in the path π

CTL Semantics - Transition System

Let $TS = (S, Act, \to, I, AP, L)$ be a transition system.

- For CTL-state-formula Φ, the *satisfaction set* $\mathit{Sat}(\Phi)$ is defined by:

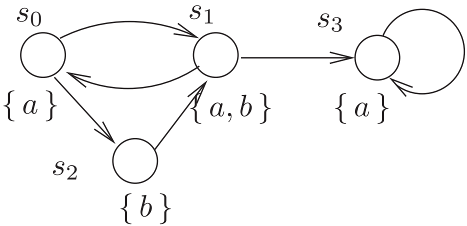
$$\mathit{Sat}(\Phi) \;=\; \{\, s \in S \mid s \models \Phi \,\}$$

- TS satisfies CTL-formula Φ iff Φ holds in all its initial states:

$$\mathit{TS} \models \Phi \quad \text{if and only if} \quad \forall s_0 \in I.\, s_0 \models \Phi$$

  This is equivalent to $I \subseteq \mathit{Sat}(\Phi)$.

CTL Semantics - Examples



$$\exists \bigcirc a$$

$$\forall \bigcirc a$$

$$\exists \Box a$$

$$\forall \Box a$$

$$\forall (a \, \mathsf{U} \, b)$$

Remark 6.10 The Semantics of Negation

TS $\not\models \Phi$ and *TS* $\not\models \neg\Phi$ is possible due to having multiple initial states, e.g., $s_0 \models \exists\Box\Phi$ and $s_0' \not\models \exists\Box\Phi$.

$$TS \not\models \exists\Box a \quad \text{and} \quad TS \not\models \neg\exists\Box a$$

6.2.3 CTL Equivalence

Definition 6.12

CTL-formulas Φ and Ψ (over AP) are *equivalent*, denoted $\Phi \equiv \Psi$ if and only if $Sat(\Phi) = Sat(\Psi)$ for *all* transition systems TS over AP.

$$\Phi \equiv \Psi \quad \text{iff} \quad (TS \models \Phi \quad \text{if and only if} \quad TS \models \Psi)$$

Duality Laws

$$
\begin{aligned}
\forall \bigcirc \Phi &\equiv \neg \exists \bigcirc \neg \Phi \\
\exists \bigcirc \Phi &\equiv \neg \forall \bigcirc \neg \Phi \\
\forall \Diamond \Phi &\equiv \neg \exists \Box \neg \Phi \\
\exists \Diamond \Phi &\equiv \neg \forall \Box \neg \Phi \\
\forall (\Phi \mathsf{U} \Psi) &\equiv \neg \exists ((\Phi \wedge \neg \Psi) \mathsf{U} (\neg \Phi \wedge \neg \Psi))
\end{aligned}
$$

Expansion Laws

Recall in LTL: $\varphi \,\mathsf{U}\, \psi \;\equiv\; \psi \;\vee\; (\varphi \;\wedge\; \bigcirc(\varphi \,\mathsf{U}\, \psi))$

In CTL:

$$
\begin{aligned}
\forall(\Phi \,\mathsf{U}\, \Psi) &\equiv \Psi \vee (\Phi \wedge \forall\bigcirc\forall(\Phi \,\mathsf{U}\, \Psi)) \\
\forall\Diamond\Phi &\equiv \Phi \vee \forall\bigcirc\forall\Diamond\Phi \\
\forall\Box\Phi &\equiv \Phi \wedge \forall\bigcirc\forall\Box\Phi
\end{aligned}
$$

$$
\begin{aligned}
\exists(\Phi \,\mathsf{U}\, \Psi) &\equiv \Psi \vee (\Phi \wedge \exists\bigcirc\exists(\Phi \,\mathsf{U}\, \Psi)) \\
\exists\Diamond\Phi &\equiv \Phi \vee \exists\bigcirc\exists\Diamond\Phi \\
\exists\Box\Phi &\equiv \Phi \wedge \exists\bigcirc\exists\Box\Phi
\end{aligned}
$$

Distributive Laws (1)

Recall in LTL: $\Box(\varphi \;\wedge\; \psi) \;\equiv\; \Box\varphi \;\wedge\; \Box\psi$ and $\Diamond(\varphi \;\vee\; \psi) \;\equiv\; \Diamond\varphi \;\vee\; \Diamond\psi$

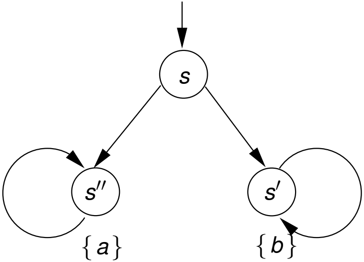
In CTL:

$$\forall\Box(\Phi \;\wedge\; \Psi) \;\equiv\; \forall\Box\Phi \;\wedge\; \forall\Box\Psi$$

$$\exists\Diamond(\Phi \vee \Psi) \;\equiv\; \exists\Diamond\Phi \;\vee\; \exists\Diamond\Psi$$

Distributive Laws (2)

Note that $\exists\Box(\Phi \wedge \Psi) \not\equiv \exists\Box\Phi \wedge \exists\Box\Psi$ and $\forall\Diamond(\Phi \vee \Psi) \not\equiv \forall\Diamond\Phi \vee \forall\Diamond\Psi$.



$s \models \forall\Diamond(a \vee b)$ since

$$s' \models a \implies s' \models a \vee b$$
$$s'' \models a \implies s'' \models a \vee b$$

However, $s \not\models \forall\Diamond a$ and $s \not\models \forall\Diamond b$.

Contents

Existential Normal Form (ENF) – Section 6.2.4

The set of CTL formulas in *existential normal form* (ENF) is given by:

$$\Phi \;::=\; \mathsf{true} \;\Big|\; a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \neg\Phi \;\Big|\; \mathsf{EX}\,\Phi \;\Big|\; \exists(\Phi_1\,\mathsf{U}\,\Phi_2) \;\Big|\; \mathsf{EG}\,\Phi$$

- For each CTL formula, there exists an equivalent CTL formula in ENF.

$$\begin{aligned} \mathsf{AX}\,\Phi &\equiv \neg\mathsf{EX}\,\neg\Phi \\ \forall(\Phi\,\mathsf{U}\,\Psi) &\equiv \neg\exists(\neg\Psi\,\mathsf{U}\,(\neg\Phi \wedge \neg\Psi)) \wedge \neg\mathsf{EG}\,\neg\Psi \end{aligned}$$

- Handle only $\mathsf{EX}\,\Phi$, $\mathsf{EG}\,\Phi$, and $\exists(\Phi_1\,\mathsf{U}\,\Phi_2)$.

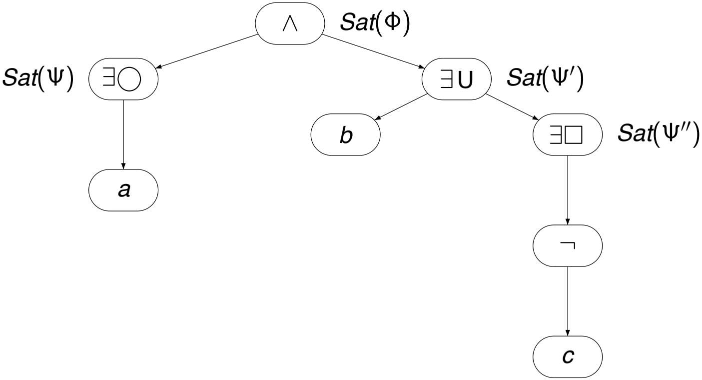
CTL Model Checking

- How to check whether *TS* satisfies CTL formula $\widehat{\Phi}$?
  - Convert the formula $\widehat{\Phi}$ into the equivalent Φ in ENF.
  - Compute *recursively* the set $Sat(\Phi) = \{ s \in S \mid s \models \Phi \}$.
  - $TS \models \Phi$ if and only if $I \subseteq Sat(\Phi)$.
- Recursive bottom-up computation of $Sat(\Phi)$:
  - Consider the parse-tree of Φ.
  - Start to compute $Sat(\Psi_i)$, for all leafs, then go one level up in the tree and determine $Sat(\cdot)$ for these nodes, repeat until the root is computed.

$$\text{e.g.,}\quad Sat(\underbrace{\Psi_1 \wedge \Psi_2}_{\text{node at level } i}) = Sat(\underbrace{\Psi_1}_{\substack{\text{node at}\\ \text{level } i+1}}) \cap Sat(\underbrace{\Psi_2}_{\substack{\text{node at}\\ \text{level } i+1}})$$

CTL Model Checking: An Example

$$\Phi \;=\; \underbrace{\mathsf{EX}\,a}_{\Psi} \;\wedge\; \underbrace{\exists(b\,\mathsf{U}\,\underbrace{\mathsf{EG}\,\neg c}_{\Psi''})}_{\Psi'} \quad .$$



Theorem 6.23 Characterization of *Sat* (1)

For all CTL formulas Φ, Ψ over *AP* it holds:

$$
\begin{array}{rcl}
Sat(\text{true}) & = & S \\
Sat(a) & = & \{\, s \in S \mid a \in L(s) \,\}, \text{ for any } a \in AP \\
Sat(\Phi \wedge \Psi) & = & Sat(\Phi) \cap Sat(\Psi) \\
Sat(\neg\Phi) & = & S \setminus Sat(\Phi) \\
Sat(\text{EX}\,\Phi) & = & \{\, s \in S \mid Post(s) \cap Sat(\Phi) \neq \emptyset \,\}
\end{array}
$$

where $TS = (S, Act, \rightarrow, I, AP, L)$ is a transition system without terminal states.

Theorem 6.23 Characterization of *Sat* (2)

- *Sat*$(\exists(\Phi \, \mathsf{U} \, \Psi))$ is the smallest subset T of S, such that:
  1. *Sat*$(\Psi) \subseteq T$ and
  2. $(s \in$ *Sat*(Φ) and *Post*$(s) \cap T \neq \emptyset)$ implies $s \in T$
- *Sat*$(\mathsf{EG} \, \Phi)$ is the largest subset T of S, such that:
  1. $T \subseteq$ *Sat*(Φ) and
  2. $s \in T$ implies *Post*$(s) \cap T \neq \emptyset$

where *TS* $= (S, \mathit{Act}, \rightarrow, I, \mathit{AP}, L)$ is a transition system without terminal states.

Algorithm 14 Computation of *Sat*

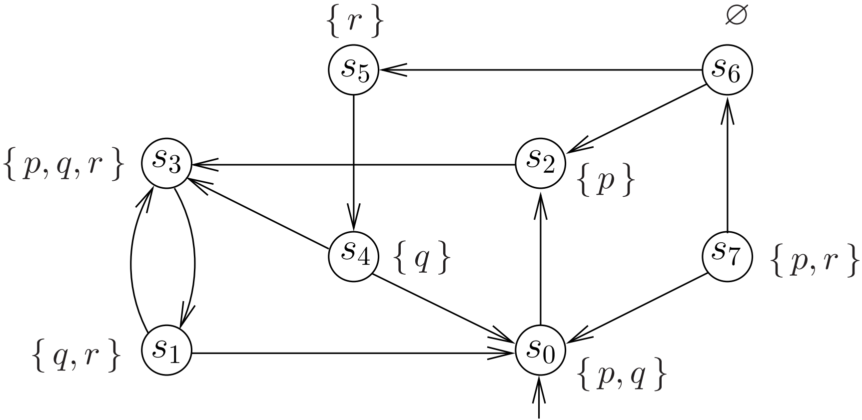
switch(Φ):

$$
\begin{array}{lcl}
a & : & \textbf{return } \{\, s \in S \mid a \in L(s) \,\}; \\
\ldots & : & \ldots\ldots \\
\mathsf{EX}\,\Psi & : & \textbf{return } \{\, s \in S \mid \mathit{Post}(s) \cap \mathit{Sat}(\Psi) \neq \emptyset \,\}; \\
\exists(\Phi_1 \,\mathsf{U}\, \Phi_2) & : & T := \mathit{Sat}(\Phi_2); \qquad \textit{compute the smallest fixed point} \\
 & & \textbf{while } \{\, s \in \mathit{Sat}(\Phi_1) \setminus T \mid \mathit{Post}(s) \cap T \neq \emptyset \,\} \neq \emptyset \textbf{ do} \\
 & & \quad \textbf{let } \; s \in \{\, s \in \mathit{Sat}(\Phi_1) \setminus T \mid \mathit{Post}(s) \cap T \neq \emptyset \,\}; \\
 & & \quad T := T \cup \{\, s \,\}; \\
 & & \textbf{od}; \\
 & & \textbf{return } T; \\
\mathsf{EG}\,\Phi & : & T := \mathit{Sat}(\Phi); \qquad \textit{compute the greatest fixed point} \\
 & & \textbf{while } \{\, s \in T \mid \mathit{Post}(s) \cap T = \emptyset \,\} \neq \emptyset \textbf{ do} \\
 & & \quad \textbf{let } \; s \in \{\, s \in T \mid \mathit{Post}(s) \cap T = \emptyset \,\}; \\
 & & \quad T := T \setminus \{\, s \,\}; \\
 & & \textbf{od}; \\
 & & \textbf{return } T;
\end{array}
$$

end switch

Computing $Sat(\exists(\Phi \,\mathsf{U}\, \Psi))$– An Example

Check EF $((p = r) \;\wedge\; (p \neq q)) \;\;\equiv\;\; \exists(true \,\mathsf{U}\, ((p = r) \;\wedge\; (p \neq q)))$



Computing $Sat(\exists(\Phi \,\mathsf{U}\, \Psi))$ – Summary

- $Sat(\exists(\Phi \,\mathsf{U}\, \Psi))$ is the smallest set $T \subseteq S$ such that

$$(1)\;\; Sat(\Psi) \subseteq T \quad \text{and} \quad (2)\;\; \{s \in T \mid s \models \Phi \wedge Post(s) \cap T \neq \emptyset\}$$

- Initially, $T_0 = \{Sat(\Psi)\}$.
- Iteratively compute

$$T_{i+1} = T_i \cup \{s \in Sat(\Phi) \mid Post(s) \cap T_i \neq \emptyset\} \text{ for } i \geq 0.$$

- In other words, computing $Sat(\exists(\Phi \,\mathsf{U}\, \Psi))$ results in

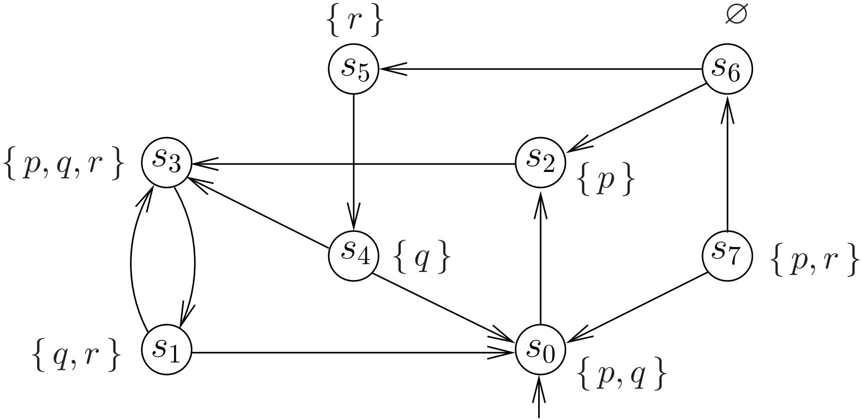
$$T_0 \subseteq T_1 \subseteq \ldots \subseteq T_j \subseteq T_{j+1} \subseteq \ldots$$

- Since we assume *TS* to be finite, there exists a $j \geq 0$ such that

$$T_j = T_{j+1} = \ldots = Sat(\exists(\Phi \,\mathsf{U}\, \Psi))$$

Computing $Sat(\mathsf{EG}\,\Phi)-$ An Example

Check $\mathsf{EG}\,q$



Computing $Sat(\mathsf{EG}\,\Phi)$ – Summary

- $Sat(\mathsf{EG}\,\Phi)$ is the largest set $T \subseteq S$ such that

$$(1)\ \ T \subseteq Sat(\Phi) \quad \text{and} \quad (2)\ \ \{s \in T \mid Post(s) \cap T \neq \emptyset\}$$

- Initially, $T_0 = Sat(\Phi)$.
- Then, iteratively compute

$$T_{i+1} = T_i \cap \{s \in Sat(\Phi) \mid Post(s) \cap T_i \neq \emptyset\}$$

- Thus, computing $Sat(\mathsf{EG}\,\Phi)$ results in

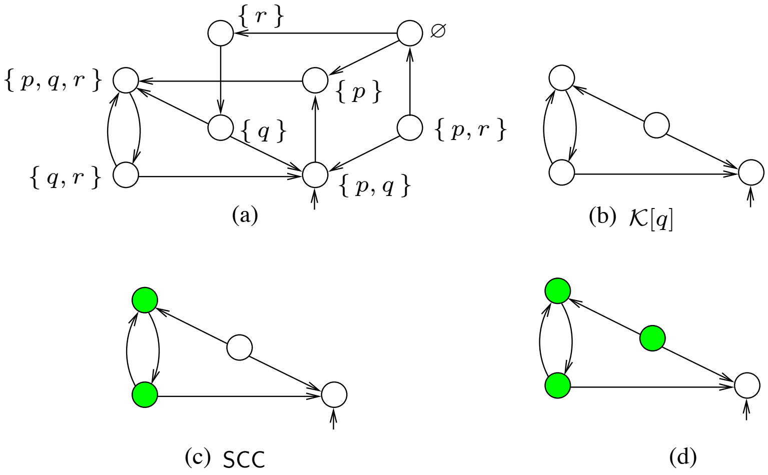
$$T_0 \supseteq T_1 \supseteq \ldots$$

- Since we assume *TS* to be finite, there exists a $j \geq 0$ such that

$$T_j = T_{j+1} = \ldots = Sat(\mathsf{EG}\,\Phi)$$

Alternative Algorithm for Computing $Sat(\mathsf{EG}\,\Phi)$

Check $\mathsf{EG}\,q$

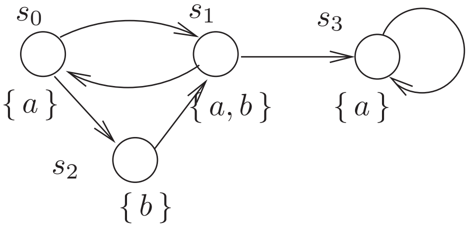


6.4.3 Time Complexity

For transition system *TS* with N states and M transitions, and CTL formula Φ, the CTL model-checking problem *TS* $\models \Phi$ can be determined in time $\mathcal{O}(|\Phi| \cdot (N + M))$.

This result applies to both algorithms for EG Φ.

CTL Semantics - Practice



$$\exists\Diamond(\exists\Box a)$$

$$\exists(a \,\mathsf{U}\, (\neg a \;\wedge\; \forall(\neg a \,\mathsf{U}\, b)))$$

Contents

6.3 Equivalence of LTL and CTL Formulas

Definition 6.17

CTL-formula Φ and LTL-formula φ (both over AP) are *equivalent*, denoted $\Phi \equiv \varphi$, if for *any* transition system TS (over AP):

$$TS \models \Phi \quad \text{if and only if} \quad TS \models \varphi$$

Theorem 6.18

Let Φ be a CTL-formula, and φ the LTL-formula obtained by eliminating all path quantifiers in Φ. Then: [Clarke & Draghicescu]

$$\Phi \equiv \varphi$$

or

there does not exist any LTL-formula that is equivalent to Φ.

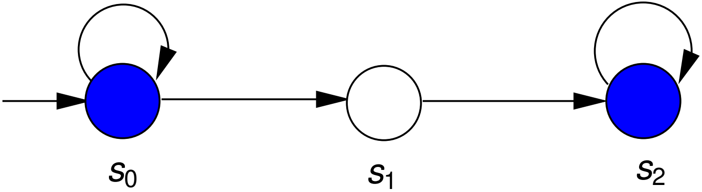
LTL and CTL are Incomparable

- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\lozenge\square a$
  - $\lozenge(a \wedge \bigcirc a)$
- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\forall\lozenge\forall\square a$
  - $\forall\lozenge(a \wedge \forall\bigcirc a)$
  - $\forall\square\exists\lozenge a$

$\Rightarrow$ Cannot be expressed = there does not exist an equivalent formula.

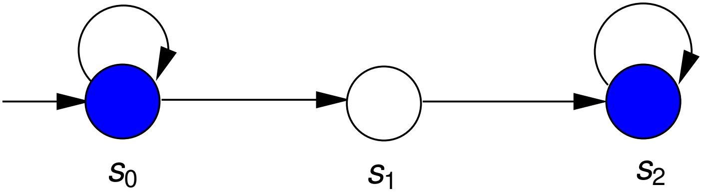
Comparing LTL and CTL (Lemma 6.19)

$$\forall\lozenge\forall\square a \not\equiv \lozenge\square a.$$



Comparing LTL and CTL (Lemma 6.19)

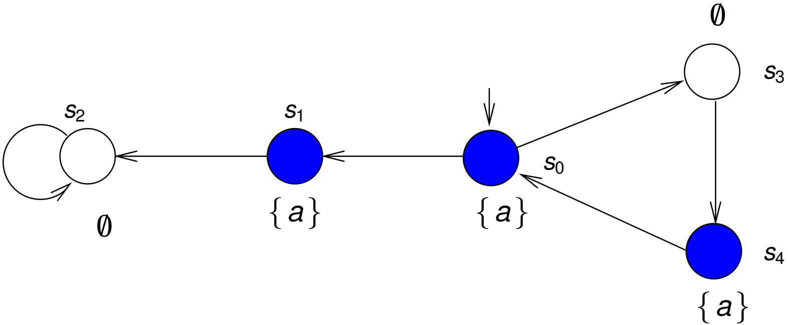
$$\forall\Diamond\forall\Box a \not\equiv \Diamond\Box a.$$



$$s_0 \models \Diamond\Box a \quad \text{but} \quad \underbrace{s_0 \not\models \forall\Diamond\forall\Box a}_{\text{path } s_0^\omega \text{ violates it}}$$

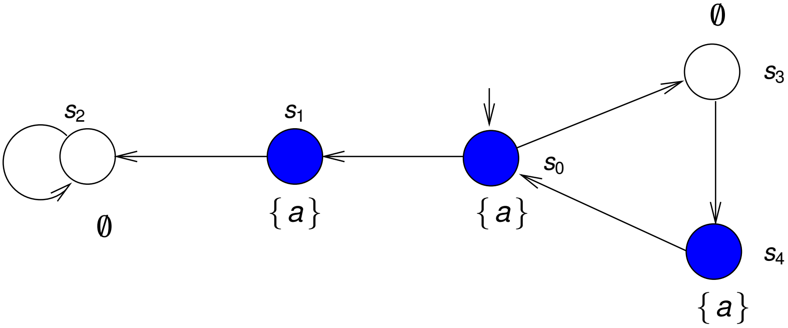
Comparing LTL and CTL (Lemma 6.20)

$$\forall\Diamond(a \wedge \forall\bigcirc a) \not\equiv \Diamond(a \wedge \bigcirc a).$$



Comparing LTL and CTL (Lemma 6.20)

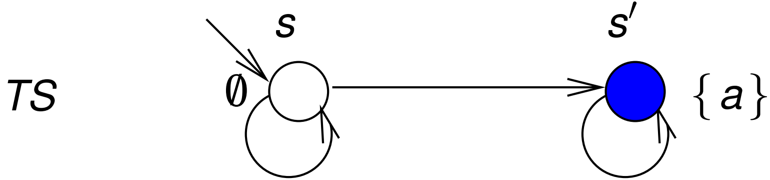
$$\forall\Diamond(a \ \wedge \ \forall\bigcirc a) \ \not\equiv \ \Diamond(a \ \wedge \ \bigcirc a).$$



$s_0 \models \Diamond(a \ \wedge \ \bigcirc a)$ but $\underbrace{s_0 \not\models \forall\Diamond(a \ \wedge \ \forall\bigcirc a)}_{\text{path } s_0\, s_1\, (s_2)^\omega \text{ violates it}}$

Comparing LTL and CTL (3)

The CTL-formula $\forall\Box\exists\Diamond a$ cannot be expressed in LTL



Linear-Time vs. Branching-Time Summary

Aspect	Linear Time	Branching Time
"behavior" in a state s	path-based: $\mathit{trace}(s)$	state-based: computation tree of s
temporal logic	LTL: path formulas φ $s \models \varphi$ iff $\forall \pi \in \rho(s).\, \pi \models \varphi$	CTL: state formulas existential path quantification $\exists\varphi$ universal path quantification: $\forall\varphi$
complexity of the model checking problems	PSPACE–complete $\mathcal{O}\left(\lvert \mathit{TS} \rvert \cdot 2^{\lvert\varphi\rvert}\right)$	PTIME $\mathcal{O}\left(\lvert \mathit{TS} \rvert \cdot \lvert\Phi\rvert\right)$
implementation-relation	trace inclusion and the like (proof is PSPACE-complete)	simulation and bisimulation (proof in polynomial time)
fairness	no special techniques	special techniques needed

Conclusion

- Branching time semantics of computation
- CTL for expressing branching time properties
- CTL model checking algorithms
- CTL and LTL are NOT comparable