4. Computation Tree Logic (1)

Huixing Fang

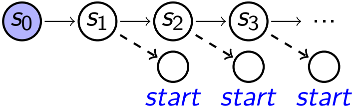
School of Information Engineering
Yangzhou University

Outline

1 Syntax

Properties for **some** or **all** computations that start in a state

1. existential path quantifier ($\exists$),
   $\exists\Diamond\Phi$: there is at least one possible computation in which a state that satisfies Φ is eventually reached
2. a universal path quantifier (denoted $\forall$),
   $\forall\Diamond\Phi$: all computations satisfy the property $\Diamond\Phi$
3. nesting universal and existential path quantifiers, $\forall\Box\exists\Diamond \mathit{start}$:



   - for every computation it is always possible to return to the initial state
   - in any state ($\Box$) of any possible computation ($\forall$), there is a possibility ($\exists$) to eventually return to the start state ($\Diamond$ start)

1 Syntax

$$\text{CTL} \begin{cases} \Phi ::= \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi & \textbf{state} \text{ formulae} \\ \varphi ::= \bigcirc\Phi \mid \Phi_2 \mathbf{U} \Phi_2 & \textbf{path} \text{ formulae} \end{cases}$$

1. state formulae express a property of a state
2. path formulae express a property of a path
3. $\exists\varphi$ holds in a state s if $\exists$ path satisfying φ that starts in s
4. $\forall\varphi$ holds in a state s if all paths that start in s satisfy φ.

$\exists\Diamond\Phi = \exists(\texttt{true}\,\mathbf{U}\,\Phi)$	Φ holds potentially
$\forall\Diamond\Phi = \forall(\texttt{true}\,\mathbf{U}\,\Phi)$	Φ is inevitable
$\exists\Box\Phi = \neg\forall\Diamond\neg\Phi$	potentially always Φ
$\forall\Box\Phi = \neg\exists\Diamond\neg\Phi$	invariantly Φ

- $\exists\neg\Diamond\neg\Phi$ is not a CTL formula
- Mutual exclusion property can be described in CTL by the formula $\forall\Box(\neg crit_1 \vee \neg crit_2)$
- The traffic light is infinitely often green: $\forall\Box\forall\Diamond green$

$\forall\Box\forall\Diamond green$ **versus** $\forall\Diamond green$

- "Every request will eventually be granted" can be described by $\forall\Box(request \rightarrow \forall\Diamond response)$

Outline

Satisfaction Relation for CTL

Let $a \in AP$, $TS = (S, Act, \rightarrow, I, AP, L)$, $s \in S$, Φ, Ψ be CTL state formulae, and φ be a CTL path formula.

1. The satisfaction relation $\models$ is defined for **state formulae** by

$$
\begin{array}{lll}
s \models a & \text{iff} & a \in L(s) \\
s \models \neg\Phi & \text{iff} & \text{not } s \models \Phi \\
s \models \Phi \wedge \Psi & \text{iff} & (s \models \Phi) \text{ and } (s \models \Psi) \\
s \models \exists\varphi & \text{iff} & \pi \models \varphi \text{ for some } \pi \in Paths(s) \\
s \models \forall\varphi & \text{iff} & \pi \models \varphi \text{ for all } \pi \in Paths(s)
\end{array}
$$

Satisfaction Relation for CTL

Let $a \in AP$, $TS = (S, Act, \rightarrow, I, AP, L)$, $s \in S$, Φ, Ψ be CTL state formulae, and φ be a CTL path formula.

1. The satisfaction relation $\models$ is defined for **state formulae** by

$$
\begin{array}{lll}
s \models a & \text{iff} & a \in L(s) \\
s \models \neg\Phi & \text{iff} & \text{not } s \models \Phi \\
s \models \Phi \wedge \Psi & \text{iff} & (s \models \Phi) \text{ and } (s \models \Psi) \\
s \models \exists\varphi & \text{iff} & \pi \models \varphi \text{ for some } \pi \in Paths(s) \\
s \models \forall\varphi & \text{iff} & \pi \models \varphi \text{ for all } \pi \in Paths(s)
\end{array}
$$

2. For path π, the satisfaction relation $\models$ for **path formulae** is

$$
\begin{array}{lll}
\pi \models \bigcirc\Phi & \text{iff} & \pi[1] \models \Phi \\
\pi \models \Phi \mathbf{U} \Psi & \text{iff} & \exists j \geq 0. \ (\pi[j] \models \Psi \wedge (\forall 0 \leq k < j. \ \pi[k] \models \Phi))
\end{array}
$$

where for path $\pi = s_0 s_1 s_2 \ldots$ and integer $k \geq 0$, $\pi[k] = s_k$

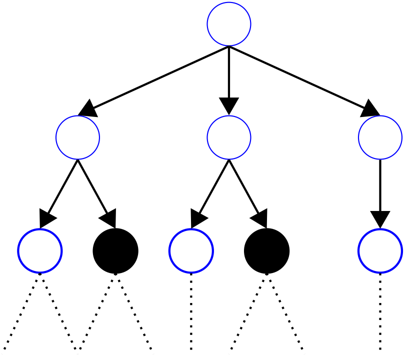
Definition 1 (CTL Semantics for Transition Systems)

Let $TS = (S, Act, \rightarrow, I, AP, L)$, the satisfaction set $Sat_{TS}(\Phi)$, or $Sat(\Phi)$, for CTL-state formula Φ is defined by:

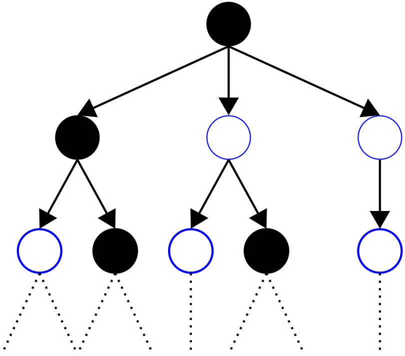
$$Sat(\Phi) = \{s \in S \mid s \models \Phi\} \ .$$

TS satisfies CTL formula Φ if and only if Φ holds in all initial states of TS:

$$TS \models \Phi \quad \text{iff} \quad \forall s_0 \in I.\ s_0 \models \Phi \quad \text{iff} \quad I \subseteq Sat(\Phi) \ .$$

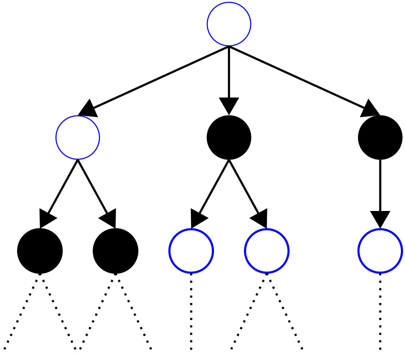


$\exists\Diamond$**black**

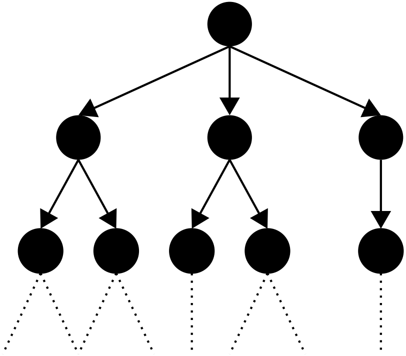


$\exists\Box$**black**

2 Semantics

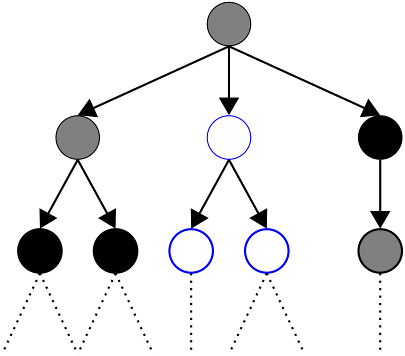


$\forall\Diamond$**black**

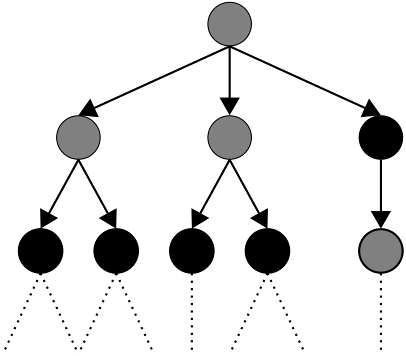


$\forall\Box$**black**

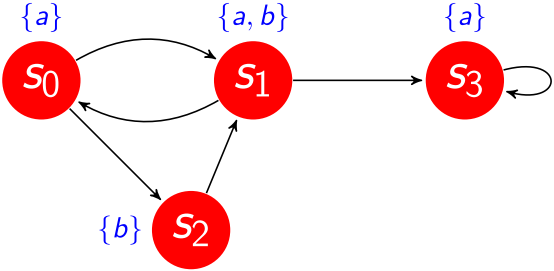
2 Semantics

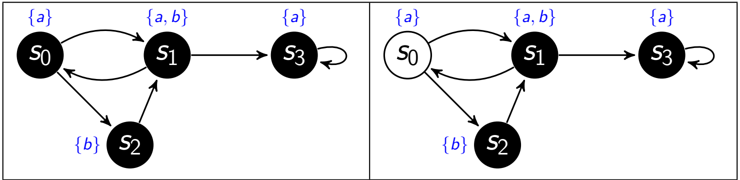


$\exists(\text{gray}\,\mathbf{U}\,\text{black})$



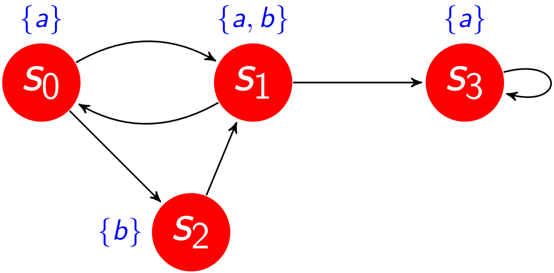
$\forall(\text{gray}\,\mathbf{U}\,\text{black})$

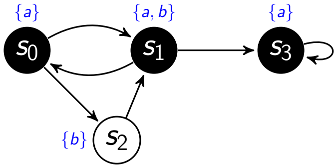
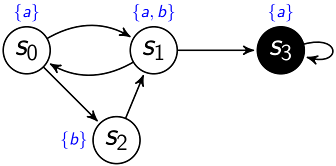




$\exists\bigcirc a$	$\forall\bigcirc a$

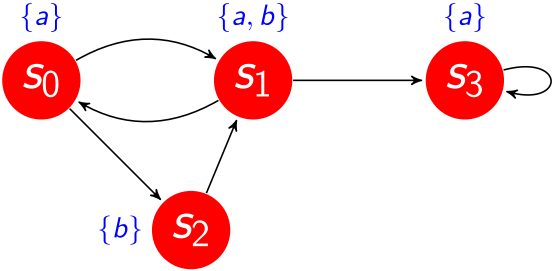
A state is colored black if the formula is valid in that state

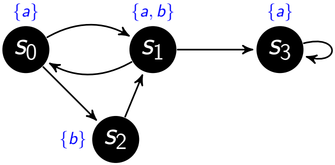
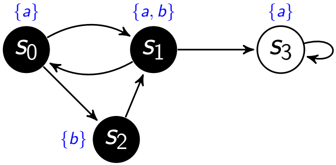


	
$\exists\Box a$	$\forall\Box a$

A state is colored black if the formula is valid in that state

2 Semantics



	
$\exists\Diamond(\exists\Box a)$	$\forall(a\,\mathbf{U}\,b)$, $\quad\exists(a\,\mathbf{U}(\neg a \land \forall(\neg a\,\mathbf{U}\,b)))$

A state is colored black if the formula is valid in that state

Let TS be a transition system and Φ a CTL formula. Is the following statement correct ?

$$\boxed{\text{if } TS \not\models \neg\Phi \text{ then } TS \models \Phi}$$

2 Semantics

Let TS be a transition system and Φ a CTL formula. Is the following statement correct ?

$$\boxed{\text{if } TS \not\models \neg\Phi \text{ then } TS \models \Phi}$$

1. $TS \models \Phi$ iff $s_0 \models \Phi$ for all initial states s_0;

Let TS be a transition system and Φ a CTL formula. Is the following statement correct ?

$$\boxed{\text{if } TS \not\models \neg\Phi \text{ then } TS \models \Phi}$$

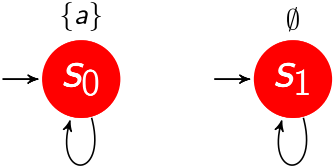
1. $TS \models \Phi$ iff $s_0 \models \Phi$ for all initial states s_0;
2. $TS \not\models \neg\Phi$ iff there exists an initial state s_0 with $s_0 \not\models \neg\Phi$ (iff there exists an initial state s_0 with $s_0 \models \Phi$)

2 Semantics

Let TS be a transition system and Φ a CTL formula. Is the following statement correct ?

$$\boxed{\text{if } TS \not\models \neg\Phi \text{ then } TS \models \Phi}$$

1. $TS \models \Phi$ iff $s_0 \models \Phi$ for all initial states s_0;
2. $TS \not\models \neg\Phi$ iff there exists an initial state s_0 with $s_0 \not\models \neg\Phi$ (iff there exists an initial state s_0 with $s_0 \models \Phi$)
3. transition system TS with 2 initial states:



$TS \not\models \exists\Box a$

$TS \not\models \neg\exists\Box a$

Weak Until

$$\varphi \mathbf{W} \psi = (\varphi \mathbf{U} \psi) \vee \Box\varphi$$

$$\exists(\Phi \mathbf{W} \Psi) = \neg\forall((\Phi \wedge \neg\Psi) \mathbf{U} (\neg\Phi \wedge \neg\Psi))$$

$$\forall(\Phi \mathbf{W} \Psi) = \neg\exists((\Phi \wedge \neg\Psi) \mathbf{U} (\neg\Phi \wedge \neg\Psi))$$

Outline

Expansion laws:

$$\exists(\Phi \, \mathbf{U} \, \Psi) \equiv \Psi \vee (\Phi \wedge \exists\bigcirc\exists(\Phi \, \mathbf{U} \, \Psi))$$

$$\forall(\Phi \, \mathbf{U} \, \Psi) \equiv \Psi \vee (\Phi \wedge \forall\bigcirc\forall(\Phi \, \mathbf{U} \, \Psi))$$

$$\exists\Diamond\Psi \equiv \Psi \vee \exists\bigcirc\exists\Diamond\Psi$$

$$\forall\Diamond\Psi \equiv \Psi \vee \forall\bigcirc\forall\Diamond\Psi$$

$$\exists(\Phi \, \mathbf{W} \, \Psi) \equiv \Psi \vee (\Phi \wedge \exists\bigcirc\exists(\Phi \, \mathbf{W} \, \Psi))$$

$$\forall(\Phi \, \mathbf{W} \, \Psi) \equiv \Psi \vee (\Phi \wedge \forall\bigcirc\forall(\Phi \, \mathbf{W} \, \Psi))$$

duality of $\Box$ and $\Diamond$:

$$\forall\Box\Phi \equiv \neg\exists\Diamond\neg\Phi$$
$$\forall\Diamond\Phi \equiv \neg\exists\Box\neg\Phi$$

self-duality of $\bigcirc$:

$$\forall\bigcirc\Phi \equiv \neg\exists\bigcirc\neg\Phi$$
$$\exists\bigcirc\Phi \equiv \neg\forall\bigcirc\neg\Phi$$

duality of $\mathbf{U}$ and $\mathbf{W}$:

$$\begin{aligned}\forall(\Phi\,\mathbf{U}\,\Psi) &\equiv \neg\exists((\Phi \wedge \neg\Psi)\,\mathbf{W}\,(\neg\Phi \wedge \neg\Psi)) \\ &\equiv \neg\exists(\neg\Psi)\,\mathbf{W}\,(\neg\Phi \wedge \neg\Psi) \\ &\equiv \neg\exists(\neg\Psi)\,\mathbf{U}\,(\neg\Phi \wedge \neg\Psi) \wedge \neg\exists\Box\neg\Psi\end{aligned}$$

Outline

4 Normal Forms for CTL

Definition 2 (Existential Normal Form for CTL)

For $a \in AP$, the set of CTL state formulae in *existential normal form* (ENF) is given by

$$\Phi ::= \texttt{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\bigcirc\Phi \mid \exists(\Phi_1 \,\mathbf{U}\, \Phi_2) \mid \exists\Box\Phi$$

Theorem 3 (Existential Normal Form for CTL)

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

Proof: Elimination of the universal path quantifier

$$\forall\bigcirc\Phi \equiv \neg\exists\bigcirc\neg\Phi,$$

$$\forall(\Phi \,\mathbf{U}\, \Psi) \equiv \neg\exists(\neg\Psi \,\mathbf{U}\, (\neg\Phi \wedge \neg\Phi)) \wedge \neg\exists\Box\neg\Psi.$$

$\Box$

The rewrite rule for $\forall\mathbf{U}$ triples the occurrences of Ψ, the translation from CTL to ENF can cause an exponential blowup

4 Normal Forms for CTL

Definition 4 (Positive Normal Form for CTL)

The set of CTL state formulae in positive normal form (PNF) is given by

1. State formulae:

$$\Phi ::= \texttt{true} \mid \texttt{flase} \mid \texttt{a} \mid \neg\texttt{a} \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid \exists\varphi \mid \forall\varphi$$

   where $a \in AP$;

2. Path formulae:

$$\varphi ::= \bigcirc\Phi \mid \Phi_1 \mathbf{U} \Phi_2 \mid \Phi_1 \mathbf{W} \Phi_2$$

4 Normal Forms for CTL

Theorem 5 (Existence of Equivalent PNF Formulae)

For each CTL formula there exists an equivalent CTL formula in PNF.

Proof: "pushing" negations "inside" the formula:

$$\neg\texttt{true} \equiv \mathit{false} \tag{1}$$

$$\neg\neg\Phi \equiv \Phi \tag{2}$$

$$\neg(\Phi \wedge \Psi) \equiv \neg\Phi \vee \neg\Psi \tag{3}$$

$$\neg\forall\bigcirc\Phi \equiv \exists\bigcirc\neg\Phi \tag{4}$$

$$\neg\exists\bigcirc\Phi \equiv \forall\bigcirc\neg\Phi \tag{5}$$

$$\neg\forall(\Phi\,\mathbf{U}\,\Psi) \equiv \exists((\Phi \wedge \neg\Psi)\,\mathbf{W}\,(\neg\Phi \wedge \neg\Psi)) \tag{6}$$

$$\neg\exists(\Phi\,\mathbf{U}\,\Psi) \equiv \forall((\Phi \wedge \neg\Psi)\,\mathbf{W}\,(\neg\Phi \wedge \neg\Psi))\ . \tag{7}$$

□

For $\forall\mathbf{U}$ and $\exists\mathbf{U}$ the number of occurrences of Ψ (and Φ) is doubled, ... may be exponentially ...

Outline

Let Φ be a CTL formula and φ an LTL formula:

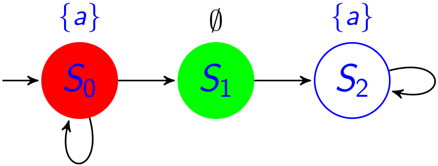
$\Phi \equiv \varphi$ iff for all transition systems TS and all states s in TS:

$$s \models_{CTL} \Phi \Leftrightarrow s \models_{LTL} \varphi \ .$$

Φ	φ
a	a
$\forall \bigcirc a$	$\bigcirc a$
$\forall (a \mathbf{U} b)$	$a \mathbf{U} b$
$\forall \Box a$	$\Box a$
$\forall \Diamond a$	$\Diamond a$
$\forall (a \mathbf{W} b)$	$a \mathbf{W} b$
$\forall \Box \forall \Diamond a$	$\Box \Diamond a$

5.1 Equivalence of CTL and LTL formulas

Transition system TS:



$TS \models_{LTL} \Diamond\Box a$

$TS \not\models_{CTL} \forall\Diamond\forall\Box a$

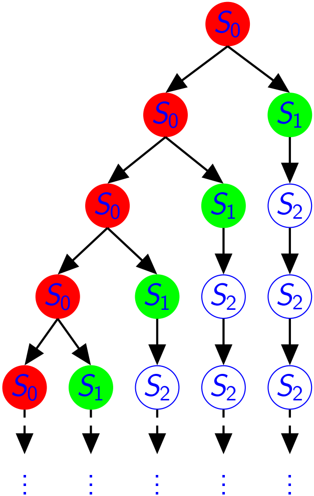
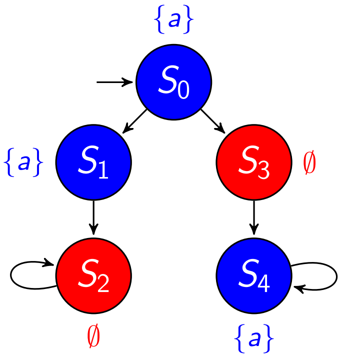


Figure: Computation tree

5.2 Expressiveness

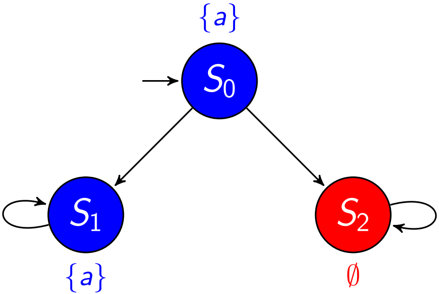
The expressive powers of LTL and CTL are incomparable

1. The CTL formulas $\forall\Diamond(a \wedge \forall\bigcirc a)$, $\forall\Diamond\forall\Box a$ and $\forall\Box\exists\Diamond a$ have no equivalent LTL formula
2. The LTL formula $\Diamond\Box a$ has no equivalent CTL formula



1. $trace(S_0 S_1 S_2^\omega) = \{a\}\{a\}\emptyset^\omega$
   $trace(S_0 S_3 S_4^\omega) = \{a\}\emptyset\{a\}^\omega$
   $\Longrightarrow$
   $TS \models_{LTL} \Diamond(a \wedge \bigcirc a)$
2. $S_0 S_1 S_2^\omega \not\models_{CTL} \Diamond(a \wedge \forall\bigcirc a)$
   $\Longrightarrow$
   $TS \not\models_{CTL} \forall\Diamond(a \wedge \forall\bigcirc a)$
3. $\Diamond(a \wedge \bigcirc a) \not\equiv \forall\Diamond(a \wedge \forall\bigcirc a)$

Does $\forall\Diamond(a \wedge \exists\bigcirc a) \equiv \Diamond(a \wedge \bigcirc a)$ hold?



$TS \not\models \Diamond(a \wedge \bigcirc a)$ $\qquad$ $TS \models \forall\Diamond(a \wedge \exists\bigcirc a)$

5.2 Expressiveness

Correct?

For each NBA $\mathcal{A}$ there is a CTL formula Φ such that for all transition systems TS :

$$TS \models \Phi \quad \text{iff} \quad Traces(TS) \subseteq \mathcal{L}_\omega(\mathcal{A})$$

Consider an NBA $\mathcal{A}$ with $\mathcal{L}_\omega(\mathcal{A}) = Words(\Diamond\Box a)$.

5.2 Expressiveness

Correct?

For each NBA $\mathcal{A}$ there is a CTL formula Φ such that for all transition systems TS :

$$TS \models \Phi \quad \text{iff} \quad Traces(TS) \subseteq \mathcal{L}_\omega(\mathcal{A})$$

Consider an NBA $\mathcal{A}$ with $\mathcal{L}_\omega(\mathcal{A}) = Words(\Diamond\Box a)$.

Correct?

If Φ is CTL formula and φ an LTL formula such that $\phi \equiv \varphi$ then $\neg\phi \equiv \neg\varphi$.

5.2 Expressiveness

Correct?

For each NBA $\mathcal{A}$ there is a CTL formula Φ such that for all transition systems TS :

$$TS \models \Phi \quad \text{iff} \quad Traces(TS) \subseteq \mathcal{L}_\omega(\mathcal{A})$$

Consider an NBA $\mathcal{A}$ with $\mathcal{L}_\omega(\mathcal{A}) = Words(\Diamond\Box a)$.

Correct?

If Φ is CTL formula and φ an LTL formula such that $\phi \equiv \varphi$ then $\neg\phi \equiv \neg\varphi$.

Consider $\Phi = \forall\Box\forall\Diamond a$, $\varphi = \Box\Diamond a$.

1. $\Phi \equiv \varphi$
2. no CTL formula that is equivalent to $\neg\varphi \equiv \Diamond\Box\neg a$

Correct?

If T_1 and T_2 are trace equivalent TS then for all CTL formulas Φ:

$$T_1 \models \Phi \quad \text{iff} \quad T_2 \models \Phi.$$

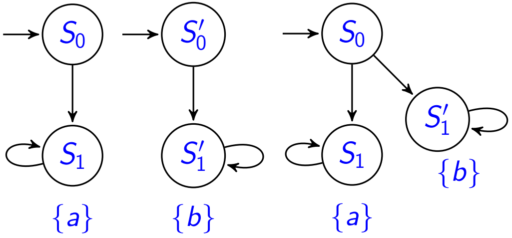


Figure: T_1 Figure: T_2

CTL formula

$$\Phi = \exists \bigcirc a \wedge \exists \bigcirc b$$

Correct?

If T_1 and T_2 are trace equivalent TS then for all CTL formulas Φ:

$$T_1 \models \Phi \quad \text{iff} \quad T_2 \models \Phi.$$

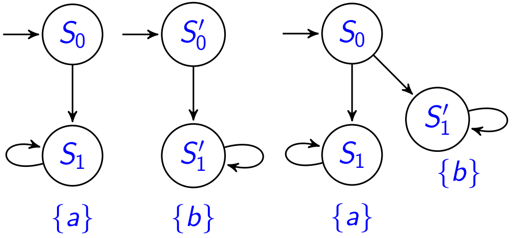


Figure: T_1 Figure: T_2

CTL formula

$$\Phi = \exists\bigcirc a \land \exists\bigcirc b$$

1. $T1$ and $T2$ are trace equivalent

Correct?

If T_1 and T_2 are trace equivalent TS then for all CTL formulas Φ:

$$T_1 \models \Phi \quad \text{iff} \quad T_2 \models \Phi.$$

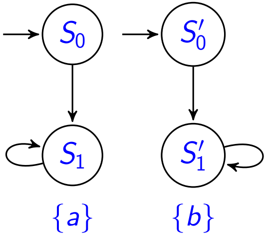


Figure: T_1

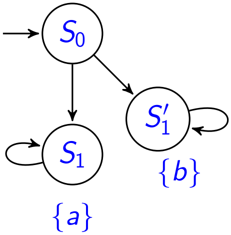


Figure: T_2

CTL formula

$$\Phi = \exists \bigcirc a \wedge \exists \bigcirc b$$

1. $T1$ and $T2$ are trace equivalent
2. $T_1 \not\models \Phi$
3. $T_2 \models \Phi$

Outline

6 CTL Model Checking

CTL-model checking can be performed by a recursive procedure

1. calculates the **satisfaction set** for all subformulae of Φ
2. and finally **checks** whether all initial states belong to this set

Consider CTL formulae in ENF:

1. $\exists\bigcirc$
2. $\exists\mathbf{U}$
3. $\exists\Box$

6 CTL Model Checking

Algorithm 1: CTL model checking (basic idea)

Input: finite transition system TS and CTL formula Φ (both over AP)

Output: $TS \models \Phi$

forall $i \leq |\Phi|$ **do**
    forall $\Psi \in Sub(\Phi)$ *with* $|\Psi| = i$ **do**
        compute $Sat(\Psi)$ from $Sat(\Psi')$; /* maximal $\Psi' \in Sub(\Psi)$ */
    end
end
return $I \subseteq Sat(\Phi)$

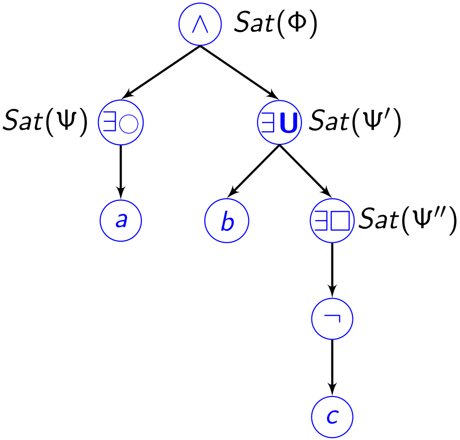
$$\boxed{Sat(\Phi) = \{s \in S \mid s \models \Phi\}}$$

The computation of $Sat(\Phi)$: **bottom-up** traversal of the parse tree of Φ:

Consider the following state formula over $AP = \{a, b, c\}$:

$$\Phi = \underbrace{\exists\bigcirc a}_{\Psi} \land \underbrace{\exists(b\,\mathbf{U}\,\underbrace{\exists\Box\neg c}_{\Psi''})}_{\Psi'}$$

1. Ψ and Ψ' are the maximal proper subformulae of Φ
2. Ψ'' is a maximal proper subformula of Ψ'



Theorem 6 (Characterization of $Sat(\cdot)$ for CTL formulae in ENF)

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system without terminal states. For all CTL formulae Φ, Ψ over AP it holds that

1. $Sat(\mathtt{true}) = S$,

Theorem 6 (Characterization of $Sat(\cdot)$ for CTL formulae in ENF)

Let $TS = (S, Act, \to, I, AP, L)$ be a transition system without terminal states. For all CTL formulae Φ, Ψ over AP it holds that

1. $Sat(\texttt{true}) = S$,
2. $Sat(a) = \{s \in S \mid a \in L(s)\}$, *for any* $a \in AP$,

6 CTL Model Checking

Theorem 6 (Characterization of $Sat(\cdot)$ for CTL formulae in ENF)

Let $TS = (S, Act, \to, I, AP, L)$ be a transition system without terminal states. For all CTL formulae Φ, Ψ over AP it holds that

1. $Sat(\texttt{true}) = S$,
2. $Sat(a) = \{s \in S \mid a \in L(s)\}$, *for any* $a \in AP$,
3. $Sat(\Phi \wedge \Psi) = Sat(\Phi) \cap Sat(\Psi)$,

6 CTL Model Checking

Theorem 6 (Characterization of $Sat(\cdot)$ for CTL formulae in ENF)

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system without terminal states. For all CTL formulae Φ, Ψ over AP it holds that

1. $Sat(\texttt{true}) = S$,
2. $Sat(a) = \{s \in S \mid a \in L(s)\}$, *for any* $a \in AP$,
3. $Sat(\Phi \wedge \Psi) = Sat(\Phi) \cap Sat(\Psi)$,
4. $Sat(\neg\Phi) = S \setminus Sat(\Phi)$,

Theorem 6 (Characterization of $Sat(\cdot)$ for CTL formulae in ENF)

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system without terminal states. For all CTL formulae Φ, Ψ over AP it holds that

1. $Sat(\texttt{true}) = S$,
2. $Sat(a) = \{s \in S \mid a \in L(s)\}$, *for any* $a \in AP$,
3. $Sat(\Phi \wedge \Psi) = Sat(\Phi) \cap Sat(\Psi)$,
4. $Sat(\neg\Phi) = S \setminus Sat(\Phi)$,
5. $Sat(\exists\bigcirc\Phi) = \{s \in S \mid Post(s) \cap Sat(\Phi) \neq \emptyset\}$,

Theorem 6 (Characterization of $Sat(\cdot)$ for CTL formulae in ENF)

Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system without terminal states. For all CTL formulae Φ, Ψ over AP it holds that

1. $Sat(\texttt{true}) = S$,
2. $Sat(a) = \{s \in S \mid a \in L(s)\}$, *for any* $a \in AP$,
3. $Sat(\Phi \wedge \Psi) = Sat(\Phi) \cap Sat(\Psi)$,
4. $Sat(\neg\Phi) = S \setminus Sat(\Phi)$,
5. $Sat(\exists\bigcirc\Phi) = \{s \in S \mid Post(s) \cap Sat(\Phi) \neq \emptyset\}$,
6. $Sat(\exists(\Phi \mathbf{U} \Psi))$ *is the* ***smallest*** *subset of* S, *such that (1)* $Sat(\Psi) \subseteq T$, *and (2)* $s \in Sat(\Phi)$ *and* $Post(s) \cap T \neq \emptyset$ *implies* $s \in T$,

Theorem 6 (Characterization of $Sat(\cdot)$ for CTL formulae in ENF)

Let $TS = (S, Act, \to, I, AP, L)$ be a transition system without terminal states. For all CTL formulae Φ, Ψ over AP it holds that

1. $Sat(\texttt{true}) = S$,
2. $Sat(a) = \{s \in S \mid a \in L(s)\}$, *for any* $a \in AP$,
3. $Sat(\Phi \wedge \Psi) = Sat(\Phi) \cap Sat(\Psi)$,
4. $Sat(\neg\Phi) = S \setminus Sat(\Phi)$,
5. $Sat(\exists\bigcirc\Phi) = \{s \in S \mid Post(s) \cap Sat(\Phi) \neq \emptyset\}$,
6. $Sat(\exists(\Phi \mathbf{U} \Psi))$ *is the* ***smallest*** *subset of* S, *such that (1)* $Sat(\Psi) \subseteq T$, *and (2)* $s \in Sat(\Phi)$ *and* $Post(s) \cap T \neq \emptyset$ *implies* $s \in T$,
7. $Sat(\exists\Box\Phi)$ *is the* ***largest*** *subset* T *of* S, *such that (3)* $T \subseteq Sat(\Phi)$ *and (4)* $s \in T$ *implies* $Post(s) \cap T \neq \emptyset$.

$Sat(\exists(\Phi\,\mathbf{U}\,\Psi))$ is the **smallest** subset of S, such that

(1) $Sat(\Psi) \subseteq T$, and

(2) $s \in Sat(\Phi)$ and $Post(s) \cap T \neq \emptyset$ implies $s \in T$

Proof: (i) Show that $T = Sat(\exists(\Phi\,\mathbf{U}\,\Psi))$ satisfies (1) and (2). From the expansion law

$$\exists(\Phi\,\mathbf{U}\,\Psi) \equiv \Psi \vee (\Phi \wedge \exists\bigcirc\exists(\Phi\,\mathbf{U}\,\Psi)),$$

it directly follows that T satisfies the properties (1) and (2).

Proof (Cont'd): (ii) Show that for **any** T satisfying properties (1) and (2) we have

$$Sat(\exists(\Phi \, \mathbf{U} \, \Psi)) \subseteq T.$$

Let $s \in Sat(\exists(\Phi \, \mathbf{U} \, \Psi))$:

1. If $s \in Sat(\Psi)$, because

   $$Sat(\Psi) \subseteq T,$$

   thus $s \in T$.

2. If $s \notin Sat(\Psi)$, there exists a path

   $$\pi = s_0 s_1 s_2 \ldots$$

   staring in $s = s_0$, such that $\pi \models \Phi \, \mathbf{U} \, \Psi$.

Proof (Cont'd): (ii.2) Let $n > 0$, such that $s_i \models \Phi, 0 \leq i \leq n$, and $s_n \models \Psi$. Then:

- $s_n \in Sat(\Psi) \subseteq T$,
- $s_{n-1} \in T$, since $s_n \in Post(s_{n-1}) \cap T$ and $s_{n-1} \in Sat(\Phi)$,
- $s_{n-2} \in T$, since $s_{n-1} \in Post(s_{n-2}) \cap T$ and $s_{n-2} \in Sat(\Phi)$,
- ......,
- $s_1 \in T$, since $s_2 \in Post(s_1) \cap T$ and $s_1 \in Sat(\Phi)$, and finally
- $s_0 \in T$, since $s_1 \in Post(s_0) \cap T$ and $s_0 \in Sat(\Phi)$.

It thus follows that $s = s_0 \in T$.

$\square$

$Sat(\exists\Box\Phi)$ is the **largest** subset T of S, such that

(1) $T \subseteq Sat(\Phi)$, and

(2) $s \in T$ implies $Post(s) \cap T \neq \emptyset$

Proof: (i) Show that $T = Sat(\exists\Box\Phi)$ satisfies (1) and (2). From the expansion law

$$\exists\Box\Phi \equiv \Phi \wedge \exists\bigcirc\exists\Box\Phi,$$

it directly follows that T satisfies the properties (1) and (2).

Proof (Cont'd): (ii) Show that for any T satisfying properties (1) and (2)

$$T \subseteq Sat(\exists\Box\Phi).$$

Let $T \subseteq S$ satisfy (1) and (2) and $s \in T$:

- $s_0 = s$.
- Since $s_0 \in T$, there exists a state $s_1 \in Post(s_0) \cap T$.
- Since $s_1 \in T$, there exists a state $s_2 \in Post(s_1) \cap T$.
- ......

Proof (Cont'd): (ii) Show that for any T satisfying properties (1) and (2)

$$T \subseteq Sat(\exists\Box\Phi).$$

Let $T \subseteq S$ satisfy (1) and (2) and $s \in T$:

- $s_0 = s$.
- Since $s_0 \in T$, there exists a state $s_1 \in Post(s_0) \cap T$.
- Since $s_1 \in T$, there exists a state $s_2 \in Post(s_1) \cap T$.
- ......

Here, property (2) is exploited in every step. From property (1), it follows that

$$s_i \in T \subseteq Sat(\Phi), \quad i \geq 0.$$

Proof (Cont'd): (ii) Show that for any T satisfying properties (1) and (2)

$$T \subseteq Sat(\exists\Box\Phi).$$

Let $T \subseteq S$ satisfy (1) and (2) and $s \in T$:

- $s_0 = s$.
- Since $s_0 \in T$, there exists a state $s_1 \in Post(s_0) \cap T$.
- Since $s_1 \in T$, there exists a state $s_2 \in Post(s_1) \cap T$.
- ......

Here, property (2) is exploited in every step. From property (1), it follows that

$$s_i \in T \subseteq Sat(\Phi), \quad i \geq 0.$$

Thus, $\pi = s_0 s_1 s_2 \ldots$ satisfies $\Box\Phi$. It follows that

$$s \in Sat(\exists\Box\Phi), \text{and then } T \subseteq Sat(\exists\Box\Phi).$$

6 CTL Model Checking

Algorithm 2: Computation of the satisfaction sets

Input: finite transition system TS with state set S, CTL formula Φ in ENF
Output: $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$

switch Φ **do**
  case `true` **do**
    return S
  case a **do**
    return $\{s \in S \mid a \in L(s)\}$
  case $\Phi_1 \wedge \Phi_2$ **do**
    return $Sat(\Phi_1) \cap Sat(\Phi_2)$
  case $\neg\Phi$ **do**
    return $S \setminus Sat(\Phi)$
  case $\exists(\Phi_1 \mathbf{U} \Phi_2)$ **do**
    return $SFP(\Phi_1, \Phi_2)$
  case $\exists\Box\Phi$ **do**
    return $GFP(\Phi_1, \Phi_2)$
end

6 CTL Model Checking

Algorithm 3: SFP

Input: CTL formulas Φ_1, Φ_2 in ENF

Output: $Sat(\exists(\Phi_1 \mathbf{U} \Phi_2))$

```
T := Sat(Φ2);
while {s ∈ Sat(Φ1) \ T | Post(s) ∩ T ≠ ∅} ≠ ∅ do
    let s ∈ {s ∈ Sat(Φ1) \ T | Post(s) ∩ T ≠ ∅};
    T := T ∪ {s};
end
return T;
```

$$
\begin{array}{l}
T := Sat(\Phi_2);\\
\textbf{while } \{s \in Sat(\Phi_1) \setminus T \mid Post(s) \cap T \neq \emptyset\} \neq \emptyset \textbf{ do}\\
\quad \text{let } s \in \{s \in Sat(\Phi_1) \setminus T \mid Post(s) \cap T \neq \emptyset\};\\
\quad T := T \cup \{s\};\\
\textbf{end}\\
\textbf{return } T;
\end{array}
$$

Algorithm 4: GFP

Input: CTL formulas Φ in ENF

Output: $Sat(\exists\Box\Phi)$

```
T := Sat(Φ);
while {s ∈ T | Post(s) ∩ T = ∅} ≠ ∅ do
    let s ∈ {s ∈ T | Post(s) ∩ T = ∅};
    T := T \ {s};
end
return T;
```

$Sat(\exists(\Phi \mathbf{U} \Psi))$ is the smallest set $T \subseteq S$, where S is the set of states in TS, such that

1. $Sat(\Psi) \subseteq T$ and
2. $(s \in Sat(\Phi)$ and $Post(s) \cap T \neq \emptyset) \implies s \in T$.

This suggests adopting the following iterative procedure:

1. $T_0 = Sat(\Psi)$ and
2. $T_{i+1} = T_i \cup \{s \in Sat(\Phi) \mid Post(s) \cap T_i \neq \emptyset\}$.

Page 348

6.1 The Until and Existential Always Operator

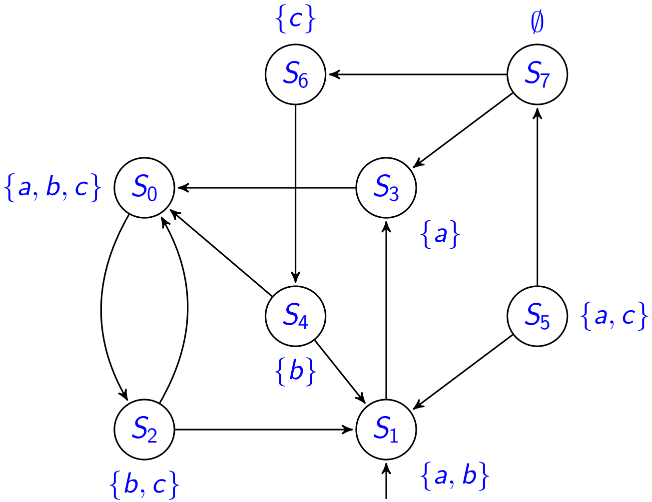
Algorithm 5: Enumerative backward search for computing $Sat(\exists(\Phi_1 \mathbf{U} \Phi_2))$

Input: finite transition system TS with state set S and CTL formula $\exists(\Phi_1 \mathbf{U} \Phi_2)$

Output: $Sat(\exists(\Phi_1 \mathbf{U} \Phi_2)) = \{s \in S \mid s \models \exists(\Phi_1 \mathbf{U} \Phi_2)\}$

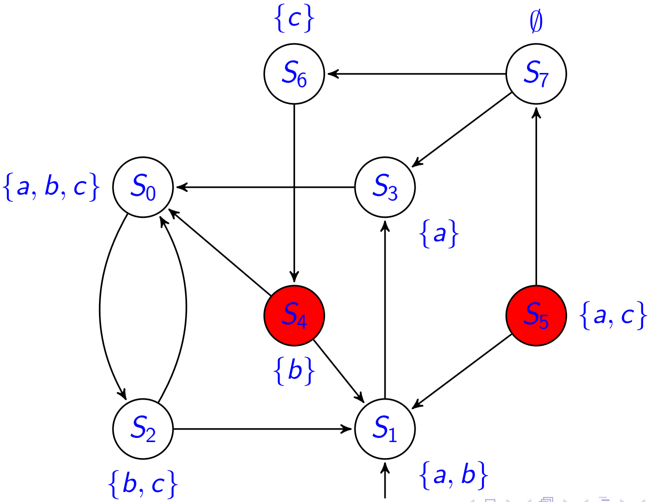
$E := Sat(\Phi_2)$; $T := E$;
while $E \neq \emptyset$ **do**
    let $s' \in E$;
    $E := E \setminus \{s'\}$;
    foreach $s \in Pre(s')$ **do**
        if $s \in Sat(\Phi_1) \setminus T$ **then**
            $E := E \cup \{s\}$;
            $T := T \cup \{s\}$;
        end
    end
end
return T;

Example of backward search for $\exists(\texttt{true}\,\mathbf{U}(\texttt{a} = \texttt{c}) \wedge (\texttt{a} \neq \texttt{b}))$

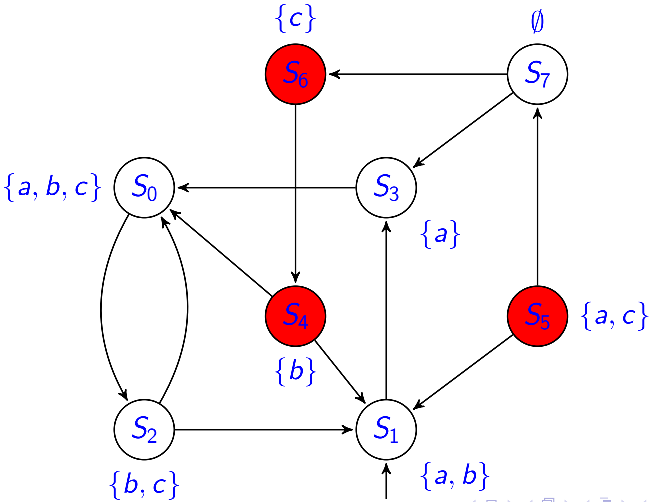


6.1 The Until and Existential Always Operator

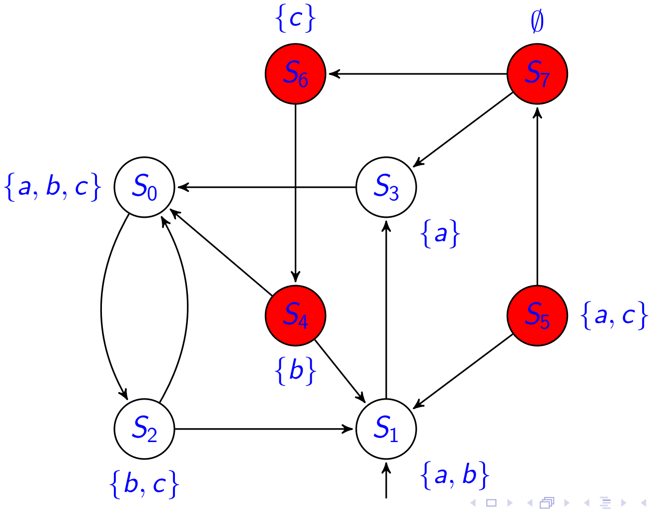
all states in the set T are colored red, we select and delete S_5 from E, but as $Pre(S_5) = \emptyset$, T remains unaffected, $T = \{S_4, S_5\}, E = \{S_4\}$



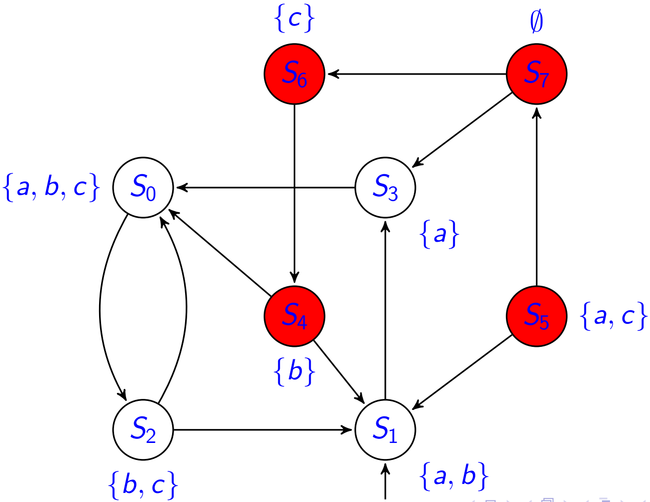
On considering $S_4 \in E$, $Pre(S_4) = \{S_6\}$ is added to T (and E), $T = \{S_4, S_5, S_6\}, E = \{S_6\}$



During the next iteration, the only predecessor of S_6 is added, $T = \{S_4, S_5, S_6, S_7\}, E = \{S_7\}$



After the fourth iteration, the algorithm terminates as there are no new predecessors of Φ-states encountered, $T = \{S_4, S_5, S_6, S_7\}, E = \{\}$



$Sat(\exists\Box\Phi)$ is the largest set $T \subseteq S$ satisfying

1. $T \subseteq Sat(\Phi)$ and
2. $s \in T$ implies $T \cap Post(s) \neq \emptyset$

The basic idea is to compute $Sat(\exists\Box\Phi)$ by means of the iteration

1. $T_0 = Sat(\Phi)$ and
2. $T_{i+1} = T_i \cap \{s \in Sat(\Phi) \mid Post(s) \cap T_i \neq \emptyset\}$

P351

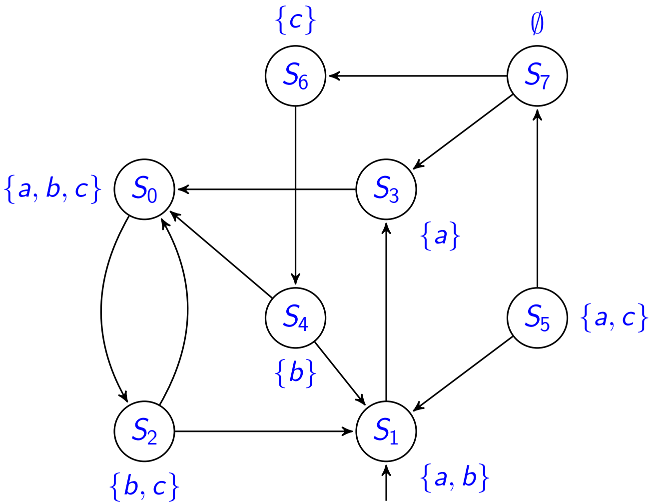
Algorithm 6: Enumerative backward search to compute $Sat(\exists\Box\Phi)$

Input: finite transition system TS with state set S and CTL formula $\exists\Box\Phi$
Output: $Sat(Sat(\exists\Box\Phi) = \{s \in S \mid s \models \exists\Box\Phi\}$

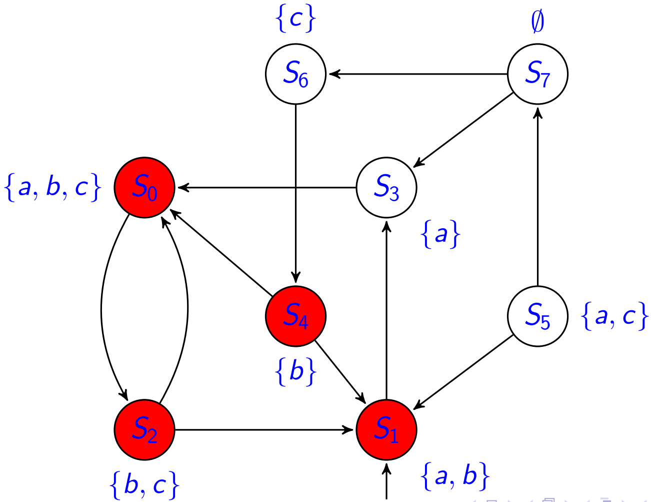
$E := S \setminus Sat(\Phi)$; $T := Sat(\Phi)$;
foreach $s \in Sat(\Phi)$ **do** $count[s] := |Post(s)|$;
while $E \neq \emptyset$ **do**
    let $s' \in E$;
    $E := E \setminus \{s'\}$;
    foreach $s \in Pre(s')$ **do**
        if $s \in T$ **then**
            $count[s] := count[s] - 1$;
            if $count[s] = 0$ **then** $T := T \setminus \{s\}$; $E := E \cup \{s\}$;
        end
    end
end
return T;

6.1 The Until and Existential Always Operator

Consider the formula $\exists\Box b$ for TS:

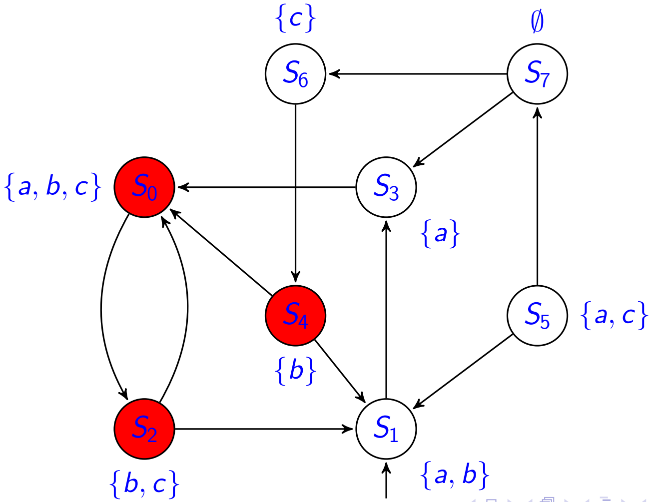


$T_0 = \{S_0, S_1, S_2, S_4\}$ and $E = \{S_3, S_5, S_6, S_7\}$ and
$count = \{S_0 \mapsto 1, S_1 \mapsto 1, S_2 \mapsto 2, S_4 \mapsto 2\}$



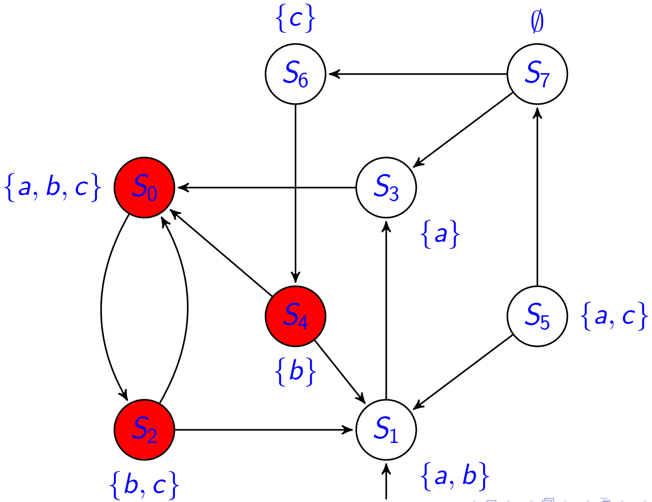
6.1 The Until and Existential Always Operator

$S3 \in E$ is selected in the first iteration. $T_1 = \{S_0, S_2, S_4\}$, $E = \{S_1, S_5, S_6, S_7\}$, $count = \{S_0 \mapsto 1, S_2 \mapsto 2, S_4 \mapsto 2\}$

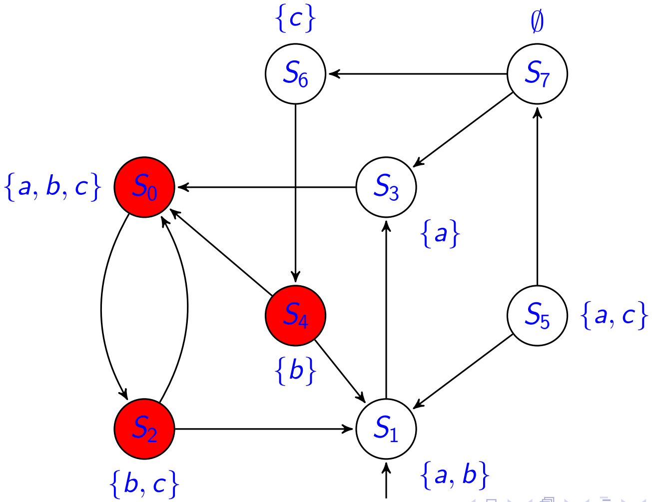


6.1 The Until and Existential Always Operator

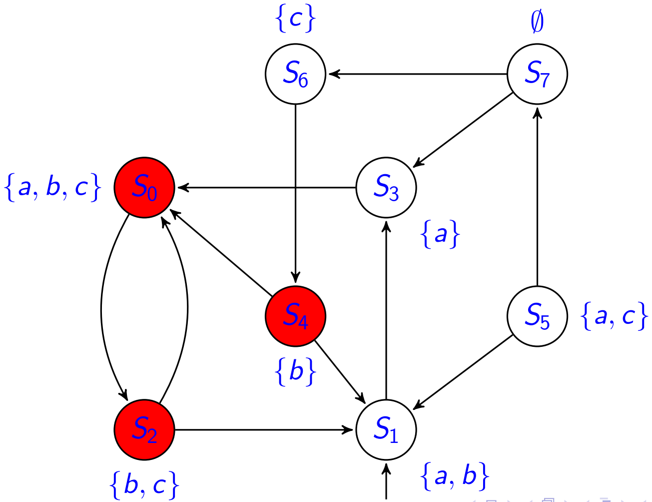
S_6 and S_7 are selected in the second and third iteration.
$T_2 = \{S_0, S_2, S_4\}$, $E = \{S_1, S_5\}$, $count = \{S_0 \mapsto 1, S_2 \mapsto 2, S_4 \mapsto 2\}$



S_1 is selected. $Pre(S_1) \cap T_2 = \{S_2, S_4\}$. $T_3 = \{S_0, S_2, S_4\}$, $E = \{S_5\}$, $count = \{S_0 \mapsto 1, S_2 \mapsto 1, S_4 \mapsto 1\}$



At last, S_5 is selected. $Pre(S_5) = \emptyset$. $T_4 = \{S_0, S_2, S_4\}$, $E = \emptyset$, $count = \{S_0 \mapsto 1, S_2 \mapsto 1, S_4 \mapsto 1\}$.



A possibility to compute $Sat(\exists\Box\Phi)$ is to only consider the Φ-states of TS and ignore all $\neg\Phi$-states

Another Way to Compute $Sat(\exists\Box\Phi)$

For $TS = (S, Act, \rightarrow, I, AP, L)$, let

$$TS[\Phi] = (S', Act, \rightarrow', I', AP, L')$$

in which, $S' = Sat(\Phi)$, $\rightarrow' = \rightarrow \cap (S' \times Act \times S')$, $I' = I \cap S'$ and $L'(s) = L(s)$ for all $s \in S'$.

1. All states in each nontrivial **strongly connected components** of $TS[\Phi]$ satisfy $\exists\Box\Phi$
2. All states in $TS[\Phi]$ that can **reach such SCC** satisfy $\exists\Box\Phi$

*A **nontrivial** SCC is an SCC that contains at least one transition.*

Theorem 7

For state s in transition system TS and CTL formula Φ:
$s \models \exists\Box\Phi$ *iff* $s \models \Phi$ *and there is a nontrivial SCC in* $TS[\phi]$ *reachable from* s.

Proof: $\Rightarrow$: Suppose $s \models \exists\Box\Phi$.

$$\Downarrow$$

According to the definition of $TS[\Phi]$, s is a state in $TS[\Phi]$.

$$\Downarrow$$

$\pi =$ path in TS starting in s and $\pi \models \Box\Phi$

$$\Downarrow$$

TS is finite, π contains a circle $c = \{s_1, ..., s_k\}$

$$\Downarrow$$

c is a SCC or contained in some SCC in $TS[\Phi]$

$$\Downarrow$$

s can reach such SCC

Proof: $\Leftarrow$:

$$\text{Suppose } s \text{ is a state in } TS[\Phi] \text{ and an SCC in } TS[\Phi] \text{ reachable from } s$$
$$\Downarrow$$
$$s' \text{ be a state in such SCC, infinite path } \pi = (s' \rightsquigarrow s')^{\omega} \models \Box\Phi$$
$$\Downarrow$$
$$\pi' = (s \rightsquigarrow_{s'} \pi) \models \Box\Phi$$
$$\Downarrow$$
$$s \models \exists\Box\Phi$$

6.2 Complexity

1. Suppose the sets of predecessors $Pre(\cdot)$ are represented as linked lists in Algorithm-5 and Algorithm-6 (Time complexity: $O(N+K)$).
2. The computation of Sat(Φ) is a bottom-up traversal over the **parse tree** of Φ and linear in $|\Phi|$

Theorem 8 (Time Complexity of CTL Model Checking)

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

6.2 Complexity

NP-complete

A decision problem C is **NP-complete** if:

1. C is in NP (that a candidate solution to C can be verified in polynomial time), and
2. Every problem in NP is reducible to C in polynomial time

Hamiltonian path

Consider the NP-complete problem of finding a Hamiltonian path in an arbitrary, connected, directed graph G

1. $G = (V, E)$
2. $V = \{v_1, ..., v_n\}$
3. $E \subseteq V \times V$

A **Hamiltonian path** is a (finite) path through the graph G which visits each state exactly once.

From G to TS

From graph G, $TS(G)$ and CTL formula Φ_n can be derived, such that

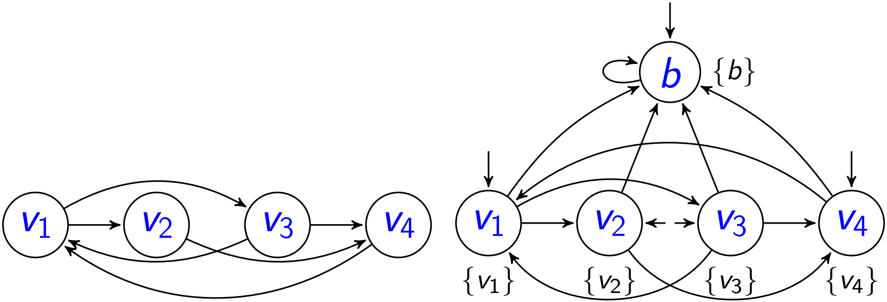
$$G \text{ contains a Hamiltonian path if and only if } TS \not\models \neg\Phi_n$$

1. $TS = (V \cup \{b\}, \{\tau\}, \rightarrow, V, V, L)$, the set initial states and set of atomic propositions are represented by V
2. $L(v_i) = \{v_i\}$, $L(b) = \emptyset$
3. The transition relation $\rightarrow$ is defined by

$$\frac{(v_i, v_j) \in E}{v_i \xrightarrow{\tau} v_j} \quad \text{and} \quad \frac{v_i \in V \cup \{b\}}{v_i \xrightarrow{\tau} b}$$

4. State b is used to ensure that TS has no terminal states

Example of encoding the Hamiltonian path problem as a transition system



(a) (b)

6.2 Complexity

Let Φ_n be defined as follows

$$\Phi_n = \bigvee_{(i_1,\ldots,i_n)\in P_m[1,\ldots,n]} \Psi(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$$

1. $\Psi(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ is a CTL formula that is satisfied if and only if $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$ is a Hamiltonian path in G.
2. The formulae $\Psi(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$ are inductively defined as follows

$$\begin{aligned} \Psi(v_i) &= v_i \\ \Psi(v_{i_1}, v_{i_2}, \ldots, v_{i_n}) &= v_{i_1} \wedge \exists \bigcirc \Psi(v_{i_2}, \ldots, v_{i_n}) \quad \text{if } n > 1 \end{aligned}$$

6.2 Complexity

An example of an instantiation of Φ_n is

$$\Phi_2 = (v_1 \wedge \exists \bigcirc v_2) \vee (v_2 \wedge \exists \bigcirc v_1)$$

and

$$\begin{aligned}\Phi_3 = {} & (v_1 \wedge \exists \bigcirc (v_2 \wedge \exists \bigcirc v_3)) \vee (v_1 \wedge \exists \bigcirc (v_3 \wedge \exists \bigcirc v_2)) \\ & \wedge (v_2 \wedge \exists \bigcirc (v_1 \wedge \exists \bigcirc v_3)) \vee (v_2 \wedge \exists \bigcirc (v_3 \wedge \exists \bigcirc v_1)) \\ & \wedge (v_3 \wedge \exists \bigcirc (v_1 \wedge \exists \bigcirc v_2)) \vee (v_3 \wedge \exists \bigcirc (v_2 \wedge \exists \bigcirc v_1)).\end{aligned}$$

$$Sat(\Psi(v_{i_1}, v_{i_2}, ..., v_{i_n})) = \begin{cases} \{v_{i_1}\} & \text{if } v_{i_1}, ..., v_{i_n} \text{ is a Hamiltonian path in } G \\ \emptyset & \text{otherwise.} \end{cases}$$

$TS \not\models \neg\Phi_n$

iff there is an initial state s of TS for which $s \not\models \neg\Phi_n$

iff there is an initial state s of TS for which $s \models \Phi_n$

iff $\exists v$ in G and a permutation $i_1, ..., i_n$ of $1, ..., n$ with $v \in Sat(\Psi(v_{i_1}, v_{i_2}, ..., v_{i_n}))$

iff $\exists v$ in G and a permutation $i_1, ..., i_n$ of $1, ..., n$, such that $v = v_{i_1}$ and $v_{i_1}, ..., v_{i_n}$ is a Hamiltonian path in G

iff G has a Hamiltonian path

6.2 Complexity

Stirling's formula

$$n! \approx \sqrt{2\pi n} \cdot (\frac{n}{e})^n \cdot (1 + \frac{1}{12n})$$

1. The size of Φ_n is exponential in the number of vertices in the graph.
2. This does not prove that there does not exist an equivalent, but shorter, CTL formula which describes the Hamiltonian path problem.
3. Shorter formalizations in CTL cannot be expected, since the CTL model-checking problem is polynomially solvable whereas the Hamiltonian path problem is NP-complete
4. Thus, if $P = NP$ then shorter $(poly(n))$ CTL formula can be found