

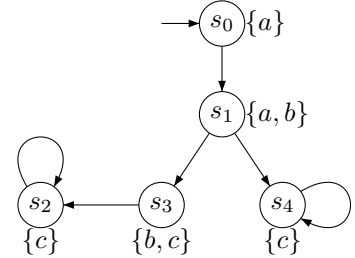
Exercise 1 (CTL Model Checking):

(4 points)

Consider the following CTL-formulas

$$\Phi_1 = \exists \Diamond \forall \Box c \quad \text{and} \quad \Phi_2 = \forall (a \cup \forall \Diamond c)$$

and the transition system outlined on the right. Decide whether $TS \models \Phi_i$ for $i = 1, 2$ using the CTL model checking algorithm from the lecture. Do not forget to translate to existential normal form and compute the satisfaction sets for subformulas.



Solution:

We apply the CTL model checking algorithm from the lecture.

- First consider the formula $\Phi_1 = \exists \Diamond \forall \Box c$:
It can be expressed equivalently in ENF:

$$\begin{aligned}
 \Phi_1 &= \exists \Diamond \forall \Box c \\
 &\equiv \exists [true \cup \forall \Box c] \\
 &\equiv \exists [true \cup \neg \exists \Diamond \neg c] \\
 &\equiv \exists [true \cup \neg \exists [true \cup \neg c]]
 \end{aligned}$$

The bottom up computation of the satisfaction sets yields:

- $Sat(true) = S$
- $Sat(c) = \{s_2, s_3, s_4\}$
- $Sat(\neg c) = \{s_0, s_1\}$
- $Sat(\exists [true \cup \neg c])$ yields a backward search as follows:
 - * $E = T = Sat(\neg c) = \{s_0, s_1\}$
 - * Choose s_0 : As $Pre(s_0) = \emptyset$, we get $E = \{s_1\}$
 - * Choose s_1 : $Pre(s_1) = \{s_0\}$. But $s_0 \notin Sat(true) \setminus T$ (i.e. it has already been visited), we get $E = \emptyset$ $\implies T = \{s_0, s_1\}$
- $Sat(\neg \exists [true \cup \neg c]) = \{s_2, s_3, s_4\}$
- $Sat(\exists [true \cup \neg \exists [true \cup \neg c]])$ again yields a backward search:
 - * $E = T = Sat(\neg \exists [true \cup \neg c]) = \{s_2, s_3, s_4\}$
 - * Choose s_2 : $Pre(s_2) = \{s_2, s_3\}$, but all predecessors are already in T . $\implies E = \{s_3, s_4\}$
 - * Choose s_3 : $Pre(s_3) = \{s_1\}$. Here we have $s_1 \notin T$ and $s_1 \in Sat(true)$. Therefore $T = T \cup \{s_1\} = \{s_1, s_2, s_3, s_4\}$ and $E = E \cup \{s_1\} = \{s_1, s_4\}$ (s_3 gets removed from E)
 - * Choose s_1 : $Pre(s_1) = \{s_0\}$. Again $s_0 \in Sat(true) \setminus T$ and therefore $T = T \cup \{s_0\} = S$ and $E = E \cup \{s_0\} = \{s_0, s_4\}$
 - * Choose s_0 : $Pre(s_0) = \emptyset$ and we directly continue with s_4 :
 - * Choose s_4 : $Pre(s_4) = \{s_1, s_4\}$ but s_1, s_4 are already in T . Therefore we stop with $E = \emptyset$.

$$\Rightarrow T = \{s_0, s_1, s_2, s_3, s_4\}$$

Therefore we have $Sat(\Phi_1) = \{s_0, s_1, s_2, s_3, s_4\}$ and $I \subseteq Sat(\Phi_1) \Rightarrow TS \models \Phi_1$

- $\Phi_2 = \forall(aU\forall\Diamond c)$. First, we transform Φ_2 into ENF:

$$\begin{aligned} \forall(aU\forall\Diamond c) &\equiv \neg\exists[(\neg\forall\Diamond c)U(\neg a \wedge \neg\forall\Diamond c)] \wedge \neg\exists\Box\neg c & (*\exists\Box\exists\Box\Psi \equiv \exists\Box\Psi*) \\ &\equiv \neg\exists[(\exists\Box\neg c)U(\neg a \wedge \exists\Box\neg c)] \wedge \neg\exists\Box\neg c \end{aligned}$$

- $Sat(a) = \{s_0, s_1\}$ and $Sat(c) = \{s_2, s_3, s_4\}$.
- $Sat(\neg a) = \{s_2, s_3, s_4\}$ and $Sat(\neg c) = \{s_0, s_1\}$.
- $Sat(\exists\Box\neg c)$ is computed by a backward search starting in the $S \setminus Sat(\neg c)$ states:
 - * $E = S \setminus Sat(\neg c) = \{s_2, s_3, s_4\}$ and $T = Sat(\neg c) = \{s_0, s_1\}$.
 - * The counter array is initialized according to TS as follows:

$$\begin{aligned} c[s_0] &= 1 & (* s_0 \text{ has only one successor} *) \\ c[s_1] &= 2 & (* s_3 \text{ and } s_4 \text{ are successors of } s_1 *) \end{aligned}$$

- * Now, choose $s_2 \in E$ and check for each of its predecessors $Pre(s_2) = \{s_2, s_3\}$ whether it belongs to T .
This is not the case, therefore we only remove s_2 .
- * Choose $s_3 \in E$: We have $s_1 \in Pre(s_3) \cap T$. Therefore with s_3 , we have found one successor of s_1 which violates $\neg c$. Therefore we decrement the number of possible successors of the state s_1

$$c[s_1] = c[s_1] - 1 = 2 - 1 = 1$$

and remove s_3 from the set E : $E = \{s_4\}$

- * Choose $s_4 \in E$: We have $Pre(s_4) = \{s_1, s_4\}$ and $s_1 \in T$. $\Rightarrow c[s_1] = c[s_1] - 1 = 0$ and $E = \emptyset$.
As $c[s_1] = 0$, s_1 is removed from T (all successor states violate $\neg c$) and included in E : $E = \{s_1\}$.
- * Choose $s_1 \in E$ with $Pre(s_1) = \{s_0\}$. As $s_0 \in T$, we decrement its counter $c[s_0] = 1 - 1 = 0$ and (because no successor state is left that could constitute a “good” path) remove it from $T = T \setminus \{s_0\} = \emptyset$. Then, s_0 is included in the set of “bad” states $E = \{s_0\}$. After another iteration, the algorithm stops with $T = \emptyset$.

The result is $Sat(\exists\Box\neg c) = T = \emptyset$

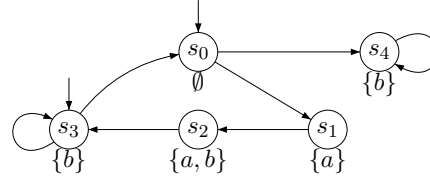
- $Sat(\neg\exists\Box\neg c) = S \setminus Sat(\exists\Box\neg c) = S$.
 - $Sat(\neg a \wedge \exists\Box\neg c) = Sat(\neg a) \cap Sat(\exists\Box\neg c) = \{s_2, s_3, s_4\} \cap \emptyset = \emptyset$
 - $Sat(\exists(\exists\Box\neg c)U(\neg a \wedge \exists\Box\neg c))$ invokes the $\exists\Psi_1 U \Psi_2$ backward search algorithm which directly terminates (as $E = Sat(\neg a \wedge \exists\Box\neg c) = \emptyset$).
 $\Rightarrow T = \emptyset$
 - $Sat(\neg\exists(\exists\Box\neg c)U(\neg a \wedge \exists\Box\neg c)) = S$
 - $Sat(\Phi_2) = Sat(\neg\exists(\exists\Box\neg c)U(\neg a \wedge \exists\Box\neg c)) \cap Sat(\neg\exists\Box\neg c) = S \cap S = S$
- $\Rightarrow I \subseteq Sat(\Phi_2) \Rightarrow TS \models \Phi_2$

Exercise 2 (CTL):

(2 points)

Consider the following CTL formulas and the transition system TS outlined on the right:

$$\begin{aligned}\Phi_1 &= \forall(aUb) \vee \exists \bigcirc (\forall \square b) \\ \Phi_2 &= \forall \square \forall(aUb) \\ \Phi_3 &= (a \wedge b) \rightarrow \exists \square \exists \bigcirc \forall(bWa) \\ \Phi_4 &= (\forall \square \exists \diamond \Phi_3)\end{aligned}$$



Give the satisfaction sets $Sat(\Phi_i)$ and decide whether $TS \models \Phi_i$ holds ($1 \leq i \leq 4$).

Solution:

For each of the CTL state formulas Φ_i , we have to compute

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

From this, we can decide $TS \models \Phi_i$ by checking $I \subseteq Sat(\Phi_i)$.

- $\Phi_1 = \forall(aUb) \vee \exists \bigcirc (\forall \square b)$:
Exemplary argumentation for s_0 :

$$\begin{aligned}s_0 \models \Phi_1 &\iff s_0 \models \forall(aUb) \vee \exists \bigcirc (\forall \square b) \\ &\iff s_0 \models \forall(aUb) \text{ or } s_0 \models \exists \bigcirc (\forall \square b)\end{aligned}$$

We have $s_0 \not\models \forall(aUb)$ because $s_0 \not\models a$ and $s_0 \not\models b$.

But $s_0 \models \exists \bigcirc (\forall \square b)$ for $\pi = s_0 s_4^\omega$: We have $\pi \models \bigcirc (\forall \square b)$.

$\implies s_0 \in Sat(\Phi_1)$.

The same line of argumentation (with $\forall(aUb)$) yields

$$Sat(\Phi_1) = \{s_0, s_1, s_2, s_3, s_4\} \text{ and } I \subseteq Sat(\Phi_1) \implies TS \models \Phi_1$$

- $\Phi_2 = \forall \square \forall(aUb)$:

$$\begin{aligned}s \models \Phi_2 &\iff \forall \pi \in Paths(s). \pi \models \square \forall(aUb) \\ &\iff \forall \pi \in Paths(s). \forall i \geq 0. \pi[i] \models \forall(aUb) \\ &\iff \forall \pi \in Paths(s). \forall i \geq 0. \forall \pi' \in Paths(\pi[i]). \pi' \models aUb\end{aligned}$$

We consider the state s_0 and the path $\pi'' = s_0 s_4^\omega$. We have $\pi'' \not\models aUb$. Choose $i = 0$ and $\pi' = \pi''$. Then the equivalences yield $s_0 \not\models \Phi_2$.

With the same arguments, we infer $s_1 \not\models \Phi_2, s_2 \not\models \Phi_2$ and $s_3 \not\models \Phi_2$.

For s_4 , we have $s_4 \models \Phi_2$ since the only possible path is $\pi' = s_4^\omega$ with $\pi' \models aUb$.

The result is:

$$Sat(\Phi_2) = \{s_4\} \implies TS \not\models \Phi_2$$

- $\Phi_3 = (a \wedge b) \rightarrow \exists \square \exists \bigcirc \forall(bWa)$:

The states s_0, s_1, s_3 and s_4 do not fulfill the state formula $(a \wedge b)$ in the premise. Therefore we have $s_0, s_1, s_3, s_4 \in Sat(\Phi_3)$.

We only have to consider the validity of the conclusion wrt. s_2 :

$$\begin{aligned}
 s_2 \models \exists \square \exists \bigcirc \forall (bWa) &\iff \exists \pi \in Paths(s_2). \pi \models \square \exists \bigcirc \forall (bWa) \\
 &\iff \exists \pi \in Paths(s_2). \forall i \geq 0. \pi[i] \models \exists \bigcirc \forall (bWa) \\
 &\implies \pi[0] = s_2 \models \exists \bigcirc \forall (bWa) \\
 &\iff \exists \pi' \in Paths(s_2). \pi' \models \bigcirc \forall (bWa) \\
 &\implies \pi'[1] = s_3 \models \forall (bWa) \\
 &\iff \forall \pi'' \in Paths(s_3). \pi'' \models bWa
 \end{aligned}$$

Consider $\pi'' = s_3 s_0 (s_4)^\omega$. Then $\pi'' \not\models bWa$ (because of s_0).
 $\implies s_2 \notin Sat(\Phi_3)$

This yields

$$Sat(\Phi_3) = \{s_0, s_1, s_3, s_4\} \text{ and } TS \models \Phi_3$$

- $\Phi_4 = (\forall \square \exists \Diamond \Phi_3)$:

Because we can reach a $Sat(\Phi_3)$ -state from every state in TS , we can directly infer

$$Sat(\Phi_4) = \{s_0, s_1, s_2, s_3, s_4\} \text{ and } TS \models \Phi_4$$

Exercise 3 (CTL and LTL expressiveness):

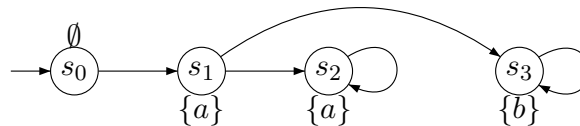
(4 points)

We consider the incomparable expressiveness of CTL and LTL.

- Using a theorem from the lecture, prove that there does not exist an equivalent LTL-formula for the CTL-formula $\Phi_1 = \forall \Diamond (a \wedge \exists \bigcirc a)$.
- Now prove directly (i.e. without the above theorem), that there does not exist an equivalent LTL-formula for the CTL-formula $\Phi_2 = \forall \Diamond \exists \bigcirc \forall \Diamond \neg a$.
Hint: Argument by contraposition, think about trace inclusion vs. CTL-equivalence!

Solution: _____

- We prove, that there is no equivalent LTL-formula for the CTL-formula $\Phi_1 = \forall \Diamond (a \wedge \exists \bigcirc a)$:
 By applying Theorem 6.18, we construct the LTL-formula $\phi_1 = \Diamond (a \wedge \bigcirc a)$ and prove, that $\Phi_1 \not\equiv \phi_1$.
 Therefore consider the following transition system TS :



We have $TS \not\models_{LTL} \varphi_1$ (because of the path $s_0 s_1 s_3^\omega$).
On the other hand, we have $TS \models_{CTL} \Phi_1$:

$$TS \models_{CTL} \forall \Diamond (a \wedge \exists \bigcirc a) \iff \forall \pi \in Paths(s_0). \exists j \geq 0. \pi[j] \models_{CTL} a \wedge \exists \bigcirc a$$

Choose $j = 1$. Then there exists a path $\pi' = s_1 s_2^\omega \in Paths(\pi[1])$ and $\pi'[1] \models a$.

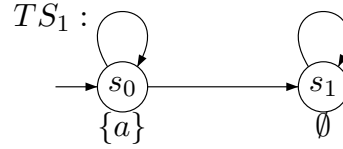
$\implies TS \models_{CTL} \Phi_1$ and $TS \not\models_{LTL} \varphi_1$

$\implies \Phi_1 \not\equiv \varphi_1$

$\implies$ (by Theorem 6.18) There exists no equivalent LTL formula for Φ_1 .

- (b) Now, we prove that there exists no equivalent LTL-formula for the CTL-formula $\Phi_2 = \forall \Diamond \exists \bigcirc \forall \Diamond \neg a$ without using Theorem 6.18:

By contradiction: Assume φ_2 an LTL-formula such that $\Phi_2 \equiv \varphi_2$. Consider the transition system TS_1 :



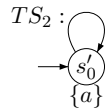
We have $Sat(\forall \Diamond \neg a) = \{s_1\}$. Therefore, we get $Sat(\exists \bigcirc \forall \Diamond \neg a) = \{s_0, s_1\}$ and trivially $Sat(\Phi_2) = \{s_0, s_1\}$.
It follows $TS_1 \models_{CTL} \Phi_2$.

Because of our assumption $\Phi_2 \equiv \varphi_2$, also $TS \models_{LTL} \varphi_2$ holds. We have

$$TS \models_{LTL} \varphi_2 \iff Traces(TS) \subseteq Words(\varphi_2) \tag{1}$$

Now consider the trace $\sigma = (\{a\})^\omega \in Traces(TS)$.

$\rightsquigarrow$ by (1) $\sigma = (\{a\})^\omega \in Words(\varphi_2)$. Consider now the following transition system TS_2 :



We have $Traces(TS_2) = \{(\{a\})^\omega\} \subseteq Words(\varphi_2)$.

$\implies TS_2 \models_{LTL} \varphi_2$.

On the other hand, we have obviously $TS_2 \not\models_{CTL} \Phi_2$ because $\neg a$ is not fulfilled by any state in TS_2 .

This yields the desired contradiction.