CS228 : Logic for Computer Science

Module 5: Temporal Logic

Lecture 26: Wrap-up, Summary, and a NuSMV Demo

Instructor: S. Akshay

IIT Bombay, India

Logistics

Pop quiz!

- **8am pop quiz!** Your task: **???**

Logistics

Course feedback

- **8am pop quiz!** Your task: **fill in the course feedback form on ASC** (deadline: **today**!)

Logistics

Course feedback

- **8am pop quiz!** Your task: **fill in the course feedback form on ASC** (deadline: **today**!)
- A second form, with questions about the TAs, will be sent out separately.

Logistics

Course feedback

- **8am pop quiz!** Your task: **fill in the course feedback form on ASC** (deadline: **today**!)
- A second form, with questions about the TAs, will be sent out separately.

End-semester exam

- **Monday, Nov 16th, 9am**

Logistics

Course feedback

- **8am pop quiz!** Your task: **fill in the course feedback form on ASC** (deadline: **today**!)
- A second form, with questions about the TAs, will be sent out separately.

End-semester exam

- **Monday, Nov 16th, 9am**

The course in numbers, and a challenge

- 26 lectures (Lecture 0 and Lectures 1–25)

Logistics

Course feedback

- **8am pop quiz!** Your task: **fill in the course feedback form on ASC** (deadline: **today**!)
- A second form, with questions about the TAs, will be sent out separately.

End-semester exam

- **Monday, Nov 16th, 9am**

The course in numbers, and a challenge

- 26 lectures (Lecture 0 and Lectures 1–25)
- 10 problem sheets

Logistics

Course feedback

- **8am pop quiz!** Your task: **fill in the course feedback form on ASC** (deadline: **today**!)
- A second form, with questions about the TAs, will be sent out separately.

End-semester exam

- **Monday, Nov 16th, 9am**

The course in numbers, and a challenge

- 26 lectures (Lecture 0 and Lectures 1–25)
- 10 problem sheets
- In addition, we will release a tough problem sheet with advanced problems, as a challenge

Logistics

Course feedback

- **8am pop quiz!** Your task: **fill in the course feedback form on ASC** (deadline: **today**!)
- A second form, with questions about the TAs, will be sent out separately.

End-semester exam

- **Monday, Nov 16th, 9am**

The course in numbers, and a challenge

- 26 lectures (Lecture 0 and Lectures 1–25)
- 10 problem sheets
- In addition, we will release a tough problem sheet with advanced problems, as a challenge
- Try it *only after* you have solved the regular problem sheets!

Topic 26.1

Wrap-up and Summary

Recap: the labelling algorithm for CTL model checking

- *Input:* a CTL formula φ and a model M. *Output:* the set of all states of M satisfying φ.

Recap: the labelling algorithm for CTL model checking

- *Input:* a CTL formula φ and a model M. *Output:* the set of all states of M satisfying φ.
- Rewrite φ using only $EX, EU, EG, \wedge, \neg, \bot$. Then label every state with the subformulas it satisfies, smallest first.

Recap: the labelling algorithm for CTL model checking

- *Input:* a CTL formula φ and a model M. *Output:* the set of all states of M satisfying φ.
- Rewrite φ using only $EX, EU, EG, \wedge, \neg, \bot$. Then label every state with the subformulas it satisfies, smallest first.
- The cases p, $\neg\psi'$ and $\psi_1 \wedge \psi_2$ are immediate. The real work is in three cases:

Recap: the labelling algorithm for CTL model checking

- *Input:* a CTL formula φ and a model M. *Output:* the set of all states of M satisfying φ.
- Rewrite φ using only $EX, EU, EG, \wedge, \neg, \bot$. Then label every state with the subformulas it satisfies, smallest first.
- The cases p, $\neg\psi'$ and $\psi_1 \wedge \psi_2$ are immediate. The real work is in three cases:

$EX\,\psi_1$

Label s if some successor of s is labelled ψ_1.

Recap: the labelling algorithm for CTL model checking

- *Input:* a CTL formula φ and a model M. *Output:* the set of all states of M satisfying φ.
- Rewrite φ using only $EX, EU, EG, \wedge, \neg, \bot$. Then label every state with the subformulas it satisfies, smallest first.
- The cases p, $\neg\psi'$ and $\psi_1 \wedge \psi_2$ are immediate. The real work is in three cases:

$EX\ \psi_1$

Label s if some successor of s is labelled ψ_1.

$E[\psi_1\ U\ \psi_2]$

Start with the ψ_2-states. Repeat till fixpoint: also label a ψ_1-state that has a successor already labelled ψ.

Recap: the labelling algorithm for CTL model checking

- *Input:* a CTL formula φ and a model M. *Output:* the set of all states of M satisfying φ.
- Rewrite φ using only $EX, EU, EG, \wedge, \neg, \bot$. Then label every state with the subformulas it satisfies, smallest first.
- The cases p, $\neg\psi'$ and $\psi_1 \wedge \psi_2$ are immediate. The real work is in three cases:

$EX\,\psi_1$

Label s if some successor of s is labelled ψ_1.

$E[\psi_1\ U\ \psi_2]$

Start with the ψ_2-states. Repeat till fixpoint: also label a ψ_1-state that has a successor already labelled ψ.

$EG\,\psi_1$

Keep only the ψ_1-states, find the strongly connected components that contain a cycle, and label every state that can reach one (backwards BFS).

Recap: the labelling algorithm for CTL model checking

- *Input:* a CTL formula φ and a model M. *Output:* the set of all states of M satisfying φ.
- Rewrite φ using only $EX, EU, EG, \wedge, \neg, \bot$. Then label every state with the subformulas it satisfies, smallest first.
- The cases p, $\neg\psi'$ and $\psi_1 \wedge \psi_2$ are immediate. The real work is in three cases:

$EX\ \psi_1$

Label s if some successor of s is labelled ψ_1.

$E[\psi_1\ U\ \psi_2]$

Start with the ψ_2-states. Repeat till fixpoint: also label a ψ_1-state that has a successor already labelled ψ.

$EG\ \psi_1$

Keep only the ψ_1-states, find the strongly connected components that contain a cycle, and label every state that can reach one (backwards BFS).

Quick recall: which of the three cases needs strongly connected components, and why?

Recap: the labelling algorithm for CTL model checking

- *Input:* a CTL formula φ and a model M. *Output:* the set of all states of M satisfying φ.
- Rewrite φ using only $EX, EU, EG, \wedge, \neg, \bot$. Then label every state with the subformulas it satisfies, smallest first.
- The cases p, $\neg\psi'$ and $\psi_1 \wedge \psi_2$ are immediate. The real work is in three cases:

EX ψ_1

Label s if some successor of s is labelled ψ_1.

$E[\psi_1 \ U \ \psi_2]$

Start with the ψ_2-states. Repeat till fixpoint: also label a ψ_1-state that has a successor already labelled ψ.

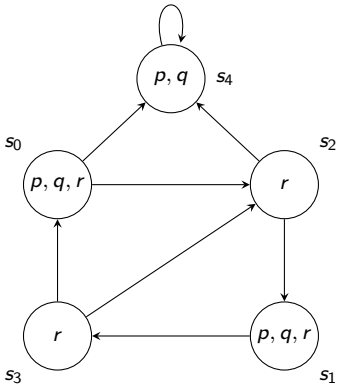
EG ψ_1

Keep only the ψ_1-states, find the strongly connected components that contain a cycle, and label every state that can reach one (backwards BFS).

Quick recall: which of the three cases needs strongly connected components, and why?

EG: an infinite path that stays among the ψ_1-states must end up going round a cycle forever.

Exercise 26.1: CTL model checking by hand

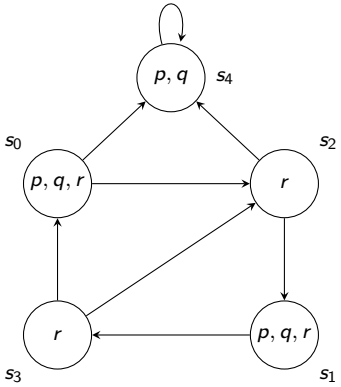


M, with initial state s_0

Let $\varphi = EG\, p \wedge E[(\neg q \vee r)\ U\ AX\, r]$, and let M be the model on the left.

(a) **Rewrite** φ as an equivalent formula φ' that uses only EX, EU, EG, $\wedge$, $\neg$ and $\bot$.

Exercise 26.1: CTL model checking by hand



M, with initial state s_0

Let $\varphi = EG\, p \wedge E[(\neg q \vee r)\ U\ AX\, r]$, and let M be the model on the left.

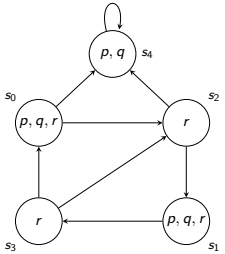
(a) **Rewrite** φ as an equivalent formula φ' that uses only EX, EU, EG, $\wedge$, $\neg$ and $\bot$.

(b) **Model check:** run the labelling algorithm on M with φ', labelling the states with every subformula of φ', smallest first. Which states satisfy φ? Does $M, s_0 \models \varphi$?

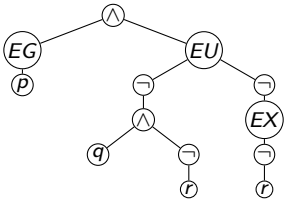
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX\, \neg r$



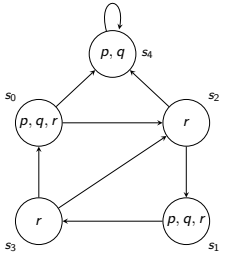
Solution to Exercise 26.1



(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$

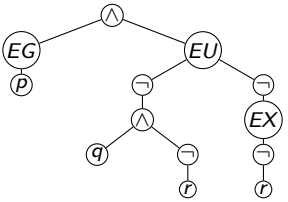


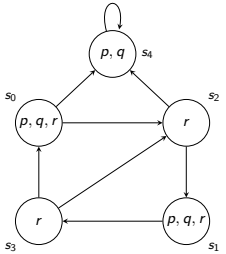
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$





(b)

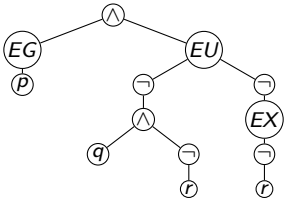
	s_0	s_1	s_2	s_3	s_4
$EG\, p$					
$\neg r$					
$q \wedge \neg r$					
$\neg(q \wedge \neg r)$					
$EX \neg r$					
$\neg EX \neg r$					
$E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$					
φ'					

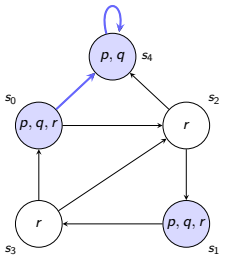
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX\, \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$					
$\neg r$					
$q \wedge \neg r$					
$\neg(q \wedge \neg r)$					
$EX\, \neg r$					
$\neg EX\, \neg r$					
$E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$					
φ'					

Step 1: $EG\, p$

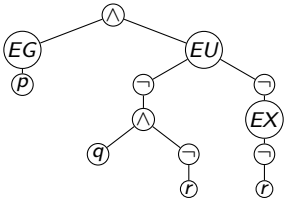
- subgraph of the p-states: s_0, s_1, s_4

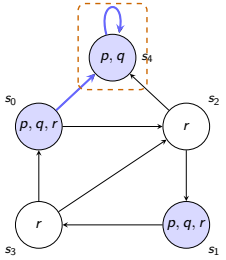
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$





(b)

	s_0	s_1	s_2	s_3	s_4
$EG\, p$					
$\neg r$					
$q \wedge \neg r$					
$\neg(q \wedge \neg r)$					
$EX \neg r$					
$\neg EX \neg r$					
$E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$					
φ'					

Step 1: $EG\, p$

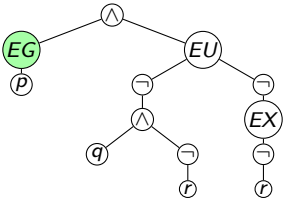
- subgraph of the p-states: s_0, s_1, s_4
- MSCC with a cycle: only $\{s_4\}$ (a self-loop); s_0, s_1 lie on no cycle

Solution to Exercise 26.1

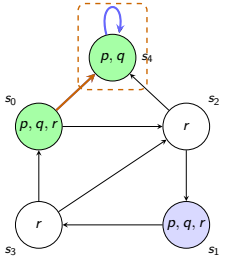
(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$



Quick recall: what if s_4 did not have a self-loop?



(b)

	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					
$q \wedge \neg r$					
$\neg(q \wedge \neg r)$					
$EX \neg r$					
$\neg EX \neg r$					
$E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$					
φ'					

Step 1: $EG\, p$

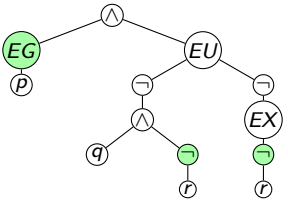
- subgraph of the p-states: s_0, s_1, s_4
- MSCC with a cycle: only $\{s_4\}$ (a self-loop); s_0, s_1 lie on no cycle
- backwards BFS from s_4 reaches s_0; s_1 cannot (s_3 is not p)

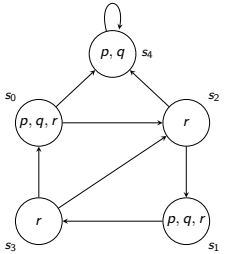
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					
$\neg(q \wedge \neg r)$					
$EX \neg r$					
$\neg EX \neg r$					
$E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$					
φ'					

Step 2: $\neg r$

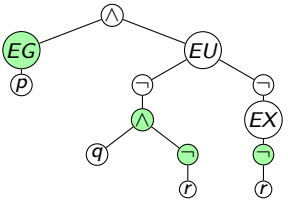
- the states not labelled r: s_4

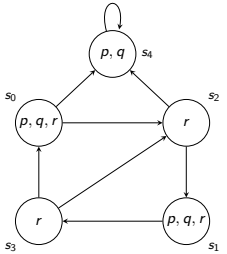
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX\, \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					✓
$\neg(q \wedge \neg r)$					
$EX\, \neg r$					
$\neg EX\, \neg r$					
$E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$					
φ'					

Step 3: $q \wedge \neg r$

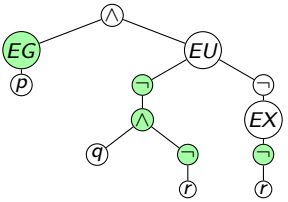
- labelled q and labelled $\neg r$: s_4

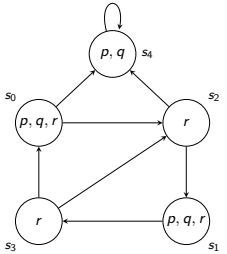
Solution to Exercise 26.1

(a) $\neg q \vee r \ \equiv \ \neg(q \wedge \neg r)$

$AX\, r \ \equiv \ \neg EX\, \neg r$

so $\varphi' \ = \ EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					✓
$\neg(q \wedge \neg r)$	✓	✓	✓	✓	
$EX\, \neg r$					
$\neg EX\, \neg r$					
$E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$					
φ'					

Step 4: $\neg(q \wedge \neg r)$

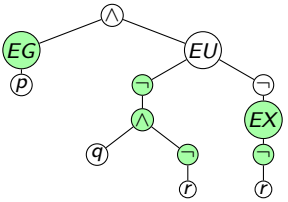
- the states not labelled $q \wedge \neg r$: s_0, s_1, s_2, s_3

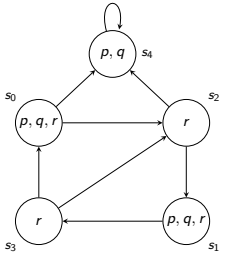
Solution to Exercise 26.1

(a) $\neg q \vee r \;\equiv\; \neg(q \wedge \neg r)$

$AX\, r \;\equiv\; \neg EX\, \neg r$

so $\varphi' \;=\; EG\, p \wedge E[\neg(q \wedge \neg r)\; U\; \neg EX\, \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					✓
$\neg(q \wedge \neg r)$	✓	✓	✓	✓	
$EX\, \neg r$	✓		✓		✓
$\neg EX\, \neg r$					
$E[\neg(q \wedge \neg r)\; U\; \neg EX\, \neg r]$					
φ'					

Step 5: $EX\, \neg r$

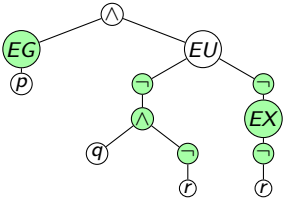
- some successor is labelled $\neg r$ (only s_4 is): s_0, s_2, s_4

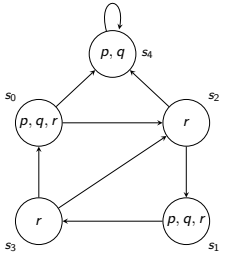
Solution to Exercise 26.1

(a) $\neg q \vee r \;\equiv\; \neg(q \wedge \neg r)$

$AX\, r \;\equiv\; \neg EX\, \neg r$

so $\varphi' \;=\; EG\, p \wedge E[\neg(q \wedge \neg r)\; U\; \neg EX\, \neg r]$





(b)

	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					✓
$\neg(q \wedge \neg r)$	✓	✓	✓	✓	
$EX\, \neg r$	✓		✓		✓
$\neg EX\, \neg r$		✓		✓	
$E[\neg(q \wedge \neg r)\; U\; \neg EX\, \neg r]$					
φ'					

Step 6: $\neg EX\, \neg r$

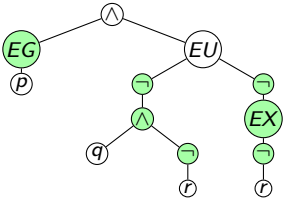
- the states not labelled $EX\, \neg r$: s_1, s_3

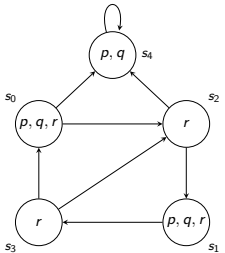
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX\, \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					✓
$\neg(q \wedge \neg r)$	✓	✓	✓	✓	
$EX\, \neg r$	✓		✓		✓
$\neg EX\, \neg r$		✓		✓	
$E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$		0		0	
φ'					

Step 7: $E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$

- round 0: labelled $\neg EX\, \neg r$: s_1, s_3

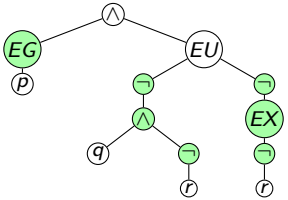
number in the U row = round in which the state is labelled

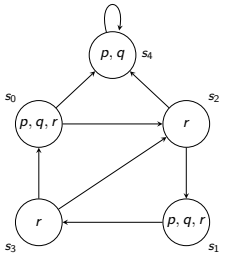
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					✓
$\neg(q \wedge \neg r)$	✓	✓	✓	✓	
$EX \neg r$	✓		✓		✓
$\neg EX \neg r$		✓		✓	
$E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$		0	1	0	
φ'					

Step 7: $E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$

- round 0: labelled $\neg EX \neg r$: s_1, s_3
- round 1: s_2 (labelled $\neg(q \wedge \neg r)$, successor s_1)

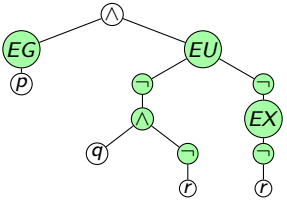
number in the U row = round in which the state is labelled

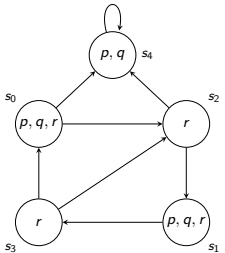
Solution to Exercise 26.1

(a) $\neg q \vee r \equiv \neg(q \wedge \neg r)$

$AX\, r \equiv \neg EX \neg r$

so $\varphi' = EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					✓
$\neg(q \wedge \neg r)$	✓	✓	✓	✓	
$EX \neg r$	✓		✓		✓
$\neg EX \neg r$		✓		✓	
$E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$	2	0	1	0	
φ'					

Step 7: $E[\neg(q \wedge \neg r)\ U\ \neg EX \neg r]$

- round 0: labelled $\neg EX \neg r$: s_1, s_3
- round 1: s_2 (labelled $\neg(q \wedge \neg r)$, successor s_1)
- round 2: s_0 (successor s_2); then nothing: s_4 is not labelled $\neg(q \wedge \neg r)$

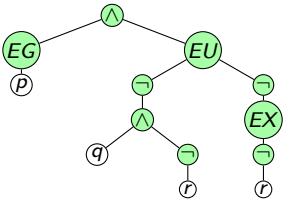
number in the U row = round in which the state is labelled

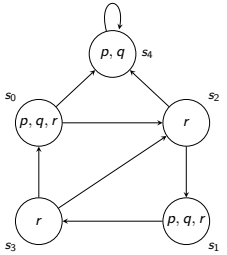
Solution to Exercise 26.1

(a) $\neg q \vee r \ \equiv \ \neg(q \wedge \neg r)$

$AX\, r \ \equiv \ \neg EX\, \neg r$

so $\varphi' \ = \ EG\, p \wedge E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$





(b)	s_0	s_1	s_2	s_3	s_4
$EG\, p$	✓				✓
$\neg r$					✓
$q \wedge \neg r$					✓
$\neg(q \wedge \neg r)$	✓	✓	✓	✓	
$EX\, \neg r$	✓		✓		✓
$\neg EX\, \neg r$		✓		✓	
$E[\neg(q \wedge \neg r)\ U\ \neg EX\, \neg r]$	2	0	1	0	
φ'	✓				

Output: $\{s_0\}$, so $M, s_0 \models \varphi$

Step 8: φ'

- labelled $EG\, p$ and the U formula: s_0

Complexity of the labelling algorithm

What is the complexity of this algo?

- All the cases except EG are clearly polynomial time.

Complexity of the labelling algorithm

What is the complexity of this algo?

- All the cases except EG are clearly polynomial time.
- EG needs strongly connected components (SCCs): this is just graph reachability, so a (backwards) BFS can do it, but carefully

Complexity of the labelling algorithm

What is the complexity of this algo?

- All the cases except EG are clearly polynomial time.
- EG needs strongly connected components (SCCs): this is just graph reachability, so a (backwards) BFS can do it, but carefully
  - never pass over a node twice: this takes linear time, $O(V + E)$

Complexity of the labelling algorithm

What is the complexity of this algo?

- All the cases except EG are clearly polynomial time.
- EG needs strongly connected components (SCCs): this is just graph reachability, so a (backwards) BFS can do it, but carefully
  - never pass over a node twice: this takes linear time, $O(V + E)$
- Altogether: $O(N \cdot (V + E))$, where N is the number of connectives in φ', V the number of states and E the number of transitions.

Complexity of the labelling algorithm

What is the complexity of this algo?

- All the cases except EG are clearly polynomial time.
- EG needs strongly connected components (SCCs): this is just graph reachability, so a (backwards) BFS can do it, but carefully
  - never pass over a node twice: this takes linear time, $O(V + E)$
- Altogether: $O(N \cdot (V + E))$, where N is the number of connectives in φ', V the number of states and E the number of transitions.

But the model itself can be huge: the state explosion problem

- M is often exponential in the number of variables and of parallel components: each extra boolean variable can double the number of states ($2^{30} \approx 10^9$)

Complexity of the labelling algorithm

What is the complexity of this algo?

- All the cases except EG are clearly polynomial time.
- EG needs strongly connected components (SCCs): this is just graph reachability, so a (backwards) BFS can do it, but carefully
  - never pass over a node twice: this takes linear time, $O(V + E)$
- Altogether: $O(N \cdot (V + E))$, where N is the number of connectives in φ', V the number of states and E the number of transitions.

But the model itself can be huge: the state explosion problem

- M is often exponential in the number of variables and of parallel components: each extra boolean variable can double the number of states ($2^{30} \approx 10^9$)

In reality, we do symbolic model checking

Use symbolic states (binary decision diagrams) instead of explicit states!

Complexity of the labelling algorithm

What is the complexity of this algo?

- All the cases except EG are clearly polynomial time.
- EG needs strongly connected components (SCCs): this is just graph reachability, so a (backwards) BFS can do it, but carefully
  - never pass over a node twice: this takes linear time, $O(V + E)$
- Altogether: $O(N \cdot (V + E))$, where N is the number of connectives in φ', V the number of states and E the number of transitions.

But the model itself can be huge: the state explosion problem

- M is often exponential in the number of variables and of parallel components: each extra boolean variable can double the number of states ($2^{30} \approx 10^9$)

In reality, we do symbolic model checking

Use symbolic states (binary decision diagrams) instead of explicit states!

Out of syllabus, but read Section 6.3 from Huth & Ryan!

Closing remarks

Model checking algo for LTL

- We have to reason on infinite paths not states!

Closing remarks

Model checking algo for LTL

- We have to reason on infinite paths not states!
- Complexity goes from PTIME (for CTL) to (believed to be) more than PTIME (for LTL). How much more?

Closing remarks

Model checking algo for LTL

- We have to reason on infinite paths not states!
- Complexity goes from PTIME (for CTL) to (believed to be) more than PTIME (for LTL). How much more?

Combining the power of CTL and LTL $\longrightarrow$ CTL^*

- Drop the CTL requirement that every temporal operator must have A/E in front!

Closing remarks

Model checking algo for LTL

- We have to reason on infinite paths not states!
- Complexity goes from PTIME (for CTL) to (believed to be) more than PTIME (for LTL). How much more?

Combining the power of CTL and LTL $\longrightarrow$ CTL^*

- Drop the CTL requirement that every temporal operator must have A/E in front!
- E.g., we can write $A[X\ p \vee X\ X\ p]$ or $E[G\ F\ p]$.

Closing remarks

Model checking algo for LTL

- We have to reason on infinite paths not states!
- Complexity goes from PTIME (for CTL) to (believed to be) more than PTIME (for LTL). How much more?

Combining the power of CTL and LTL $\longrightarrow$ CTL^*

- Drop the CTL requirement that every temporal operator must have A/E in front!
- E.g., we can write $A[X\ p \vee X\ X\ p]$ or $E[G\ F\ p]$.
- **Exercise 26.2**
  1. Show that LTL and CTL are strict subsets of CTL^*.

Closing remarks

Model checking algo for LTL

- We have to reason on infinite paths not states!
- Complexity goes from PTIME (for CTL) to (believed to be) more than PTIME (for LTL). How much more?

Combining the power of CTL and LTL $\longrightarrow$ CTL^*

- Drop the CTL requirement that every temporal operator must have A/E in front!
- E.g., we can write $A[X\ p \vee X\ X\ p]$ or $E[G\ F\ p]$.
- **Exercise 26.2**
  1. Show that LTL and CTL are strict subsets of CTL^*.
  2. But is it more than their union?

Closing remarks

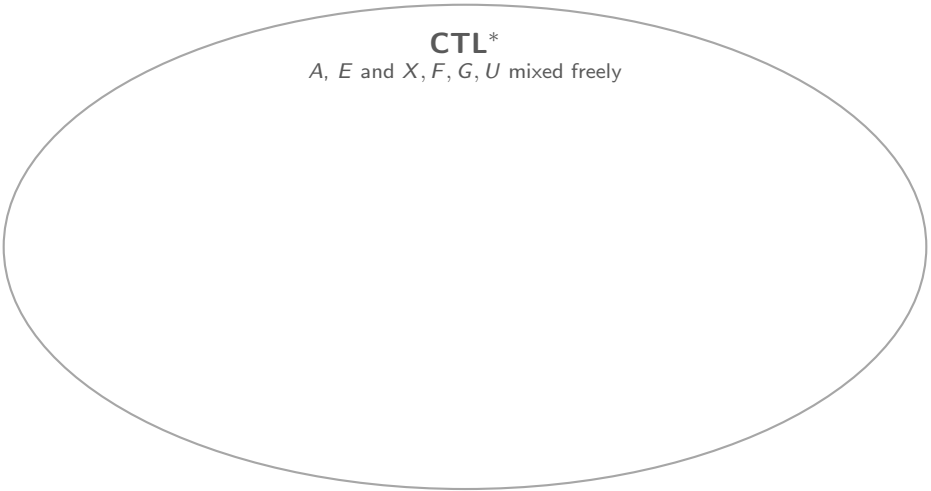
Model checking algo for LTL

- We have to reason on infinite paths not states!
- Complexity goes from PTIME (for CTL) to (believed to be) more than PTIME (for LTL). How much more?

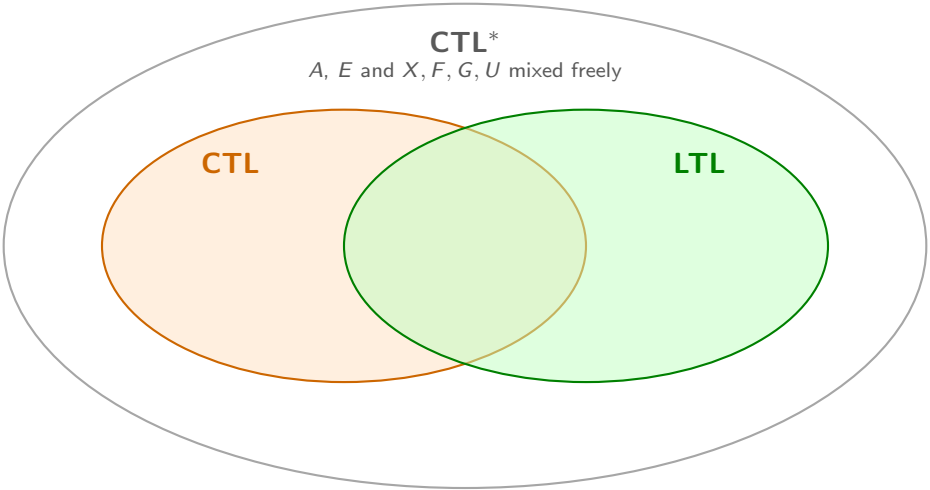
Combining the power of CTL and LTL $\longrightarrow$ CTL^*

- Drop the CTL requirement that every temporal operator must have A/E in front!
- E.g., we can write $A[X\ p \vee X\ X\ p]$ or $E[G\ F\ p]$.
- **Exercise 26.2**
  1. Show that LTL and CTL are strict subsets of CTL^*.
  2. But is it more than their union? Prove or disprove: there exists φ in CTL^* which cannot be expressed in LTL or CTL!

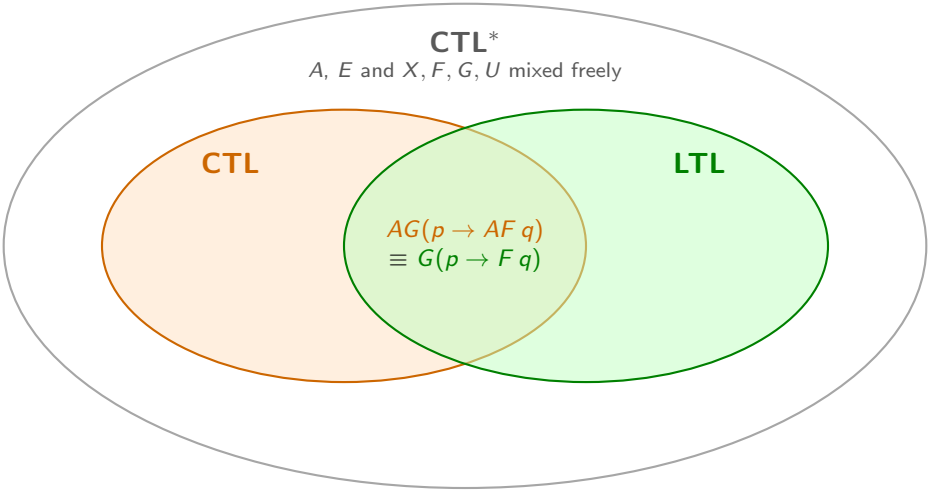
Expressive power: CTL, LTL and CTL*



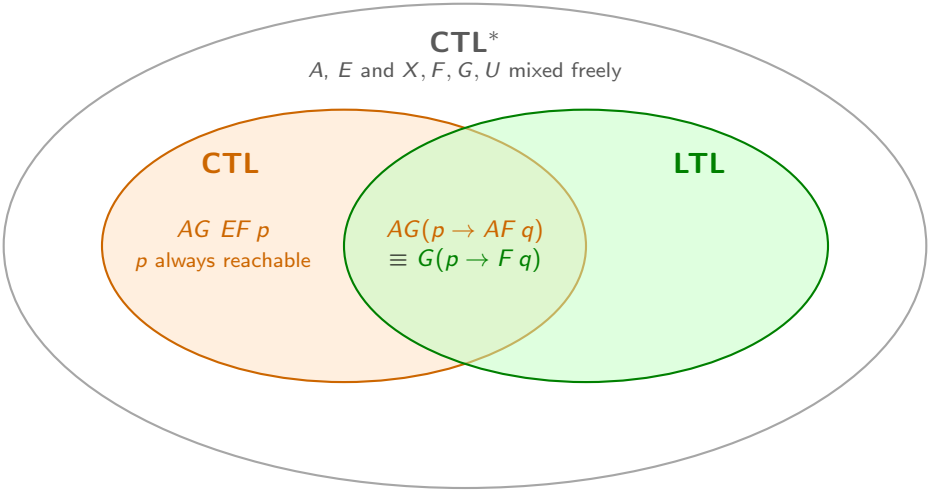
Expressive power: CTL, LTL and CTL*



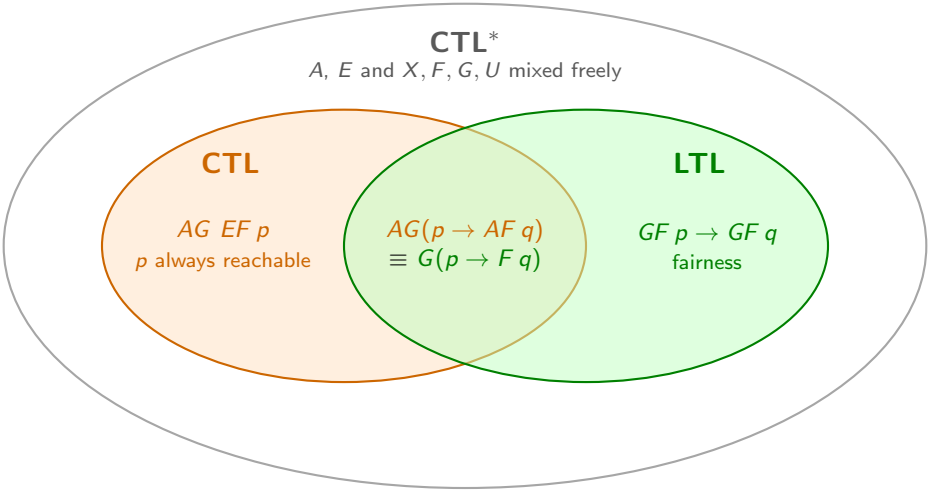
Expressive power: CTL, LTL and CTL*



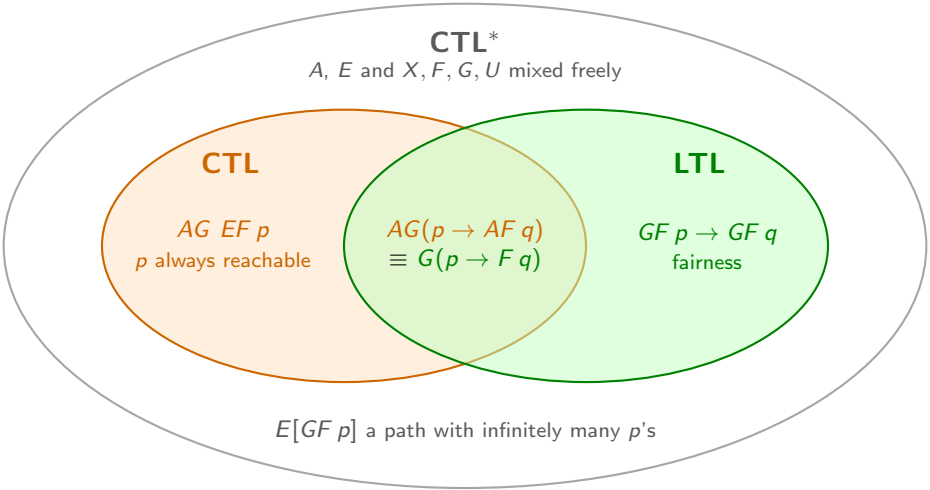
Expressive power: CTL, LTL and CTL*



Expressive power: CTL, LTL and CTL*



Expressive power: CTL, LTL and CTL*



Summary

Linear-time $\rightarrow$ LTL

- evaluated on *paths*; $M, s \models \varphi$ means *all* paths from s satisfy φ
- connectives X, F, G, U, e.g., $G(\textbf{req} \rightarrow F\ \textbf{grant})$
- F, G are duals, but $F(\varphi \wedge \psi) \not\equiv F\varphi \wedge F\psi$
- adequate set: $\{\top, \neg, \wedge, X, U\}$

Summary

Linear-time → LTL

- evaluated on *paths*; $M, s \models \varphi$ means *all* paths from s satisfy φ
- connectives X, F, G, U, e.g., $G(\textbf{req} \rightarrow F\ \textbf{grant})$
- F, G are duals, but $F(\varphi \wedge \psi) \not\equiv F\varphi \wedge F\psi$
- adequate set: $\{\top, \neg, \wedge, X, U\}$

Branching-time → CTL

- LTL cannot assert that a path *exists*: add quantifiers A (all paths) and E (some path)
- each temporal connective comes with one, e.g., $AG(\textbf{req} \rightarrow AF\ \textbf{grant})$, $AG\ EF\ \textbf{ready}$
- evaluated on *states* of the transition system
- adequate set: $\{AU, EU, EX\}$

Summary

Linear-time → LTL

- evaluated on *paths*; $M, s \models \varphi$ means *all* paths from s satisfy φ
- connectives X, F, G, U, e.g., $G(\textbf{req} \rightarrow F\ \textbf{grant})$
- F, G are duals, but $F(\varphi \wedge \psi) \not\equiv F\varphi \wedge F\psi$
- adequate set: $\{\top, \neg, \wedge, X, U\}$

Branching-time → CTL

- LTL cannot assert that a path *exists*: add quantifiers A (all paths) and E (some path)
- each temporal connective comes with one, e.g., $AG(\textbf{req} \rightarrow AF\ \textbf{grant})$, $AG\ EF\ \textbf{ready}$
- evaluated on *states* of the transition system
- adequate set: $\{AU, EU, EX\}$

Model checking

- does $M, s_0 \models \varphi$ hold? For CTL we compute *all* states satisfying φ
- CTL: rewrite with EX, EU, EG, then label states with subformulas, bottom-up; time $O(N \cdot (V + E))$ (N connectives, V states, E transitions)
- LTL: infinite paths; believed to be more than PTIME, while CTL is in PTIME

Summary

Linear-time → LTL

- evaluated on *paths*; $M, s \models \varphi$ means *all* paths from s satisfy φ
- connectives X, F, G, U, e.g., $G(\textbf{req} \rightarrow F\ \textbf{grant})$
- F, G are duals, but $F(\varphi \wedge \psi) \not\equiv F\varphi \wedge F\psi$
- adequate set: $\{\top, \neg, \wedge, X, U\}$

Branching-time → CTL

- LTL cannot assert that a path *exists*: add quantifiers A (all paths) and E (some path)
- each temporal connective comes with one, e.g., $AG(\textbf{req} \rightarrow AF\ \textbf{grant})$, $AG\ EF$ **ready**
- evaluated on *states* of the transition system
- adequate set: $\{AU, EU, EX\}$

Model checking

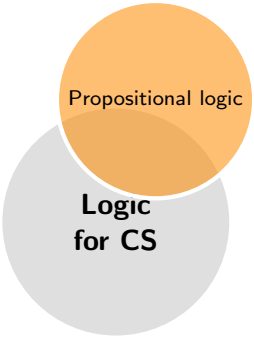
- does $M, s_0 \models \varphi$ hold? For CTL we compute *all* states satisfying φ
- CTL: rewrite with EX, EU, EG, then label states with subformulas, bottom-up; time $O(N \cdot (V + E))$ (N connectives, V states, E transitions)
- LTL: infinite paths; believed to be more than PTIME, while CTL is in PTIME

Neither subsumes the other: $AG\ EF\ p$ is CTL only, $GF\ p \rightarrow GF\ q$ is LTL only

Summary of the course



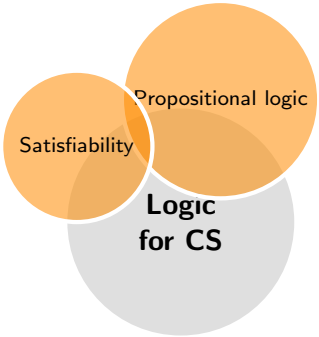
Summary of the course



Propositional logic
- syntax and semantics
- natural deduction
- normal forms

logic puzzles, checking arguments

Summary of the course



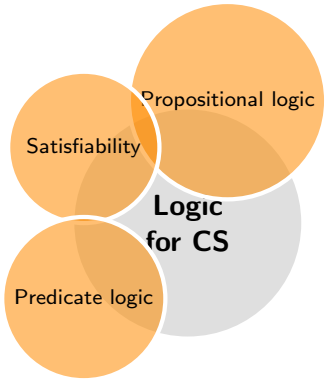
Propositional logic
- syntax and semantics
- natural deduction
- normal forms

logic puzzles, checking arguments

Satisfiability
- resolution
- DPLL and CDCL
- SAT solver demo

HW/SW verification, planning

Summary of the course



Propositional logic
- syntax and semantics
- natural deduction
- normal forms

logic puzzles, checking arguments

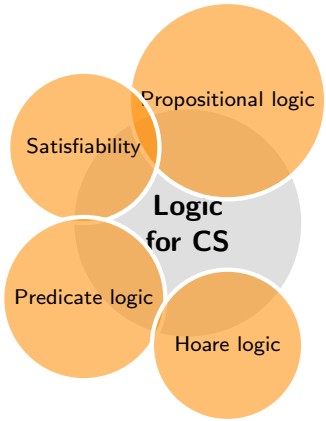
Satisfiability
- resolution
- DPLL and CDCL
- SAT solver demo

HW/SW verification, planning

Predicate logic
- syntax and semantics
- natural deduction
- limits of FOL, DPLL(T)

databases, SMT solvers

Summary of the course



Propositional logic
- syntax and semantics
- natural deduction
- normal forms

logic puzzles, checking arguments

Satisfiability
- resolution
- DPLL and CDCL
- SAT solver demo

HW/SW verification, planning

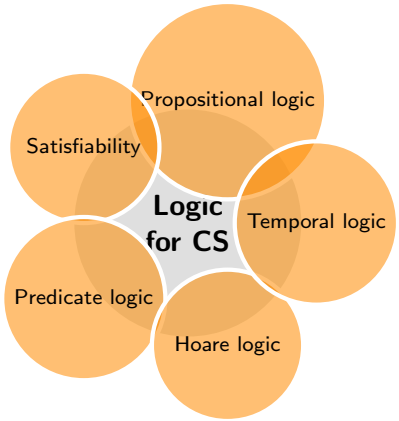
Predicate logic
- syntax and semantics
- natural deduction
- limits of FOL, DPLL(T)

databases, SMT solvers

Hoare logic
- triples and proof rules
- invariants and variants
- total correctness

verifying code

Summary of the course



Propositional logic
- syntax and semantics
- natural deduction
- normal forms

logic puzzles, checking arguments

Satisfiability
- resolution
- DPLL and CDCL
- SAT solver demo

HW/SW verification, planning

Predicate logic
- syntax and semantics
- natural deduction
- limits of FOL, DPLL(T)

databases, SMT solvers

Hoare logic
- triples and proof rules
- invariants and variants
- total correctness

verifying code

Temporal logic
- LTL and CTL
- expressive power
- CTL model checking

reactive systems (OS, lift controller, pacemaker)

Logic: the language and the calculus for reasoning

Topic 26.2

Model checking demo (not in syllabus)