

CS 267: Automated Verification

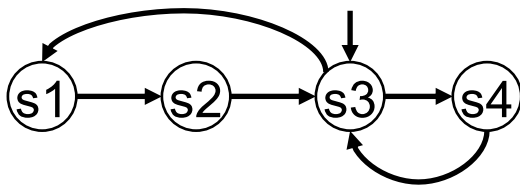
Lecture 2: Linear vs. Branching time. Temporal Logics: CTL, CTL*. CTL model checking algorithm. Counter-example generation.

Instructor: Tevfik Bultan

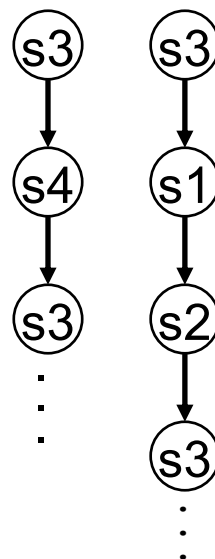
Linear Time vs. Branching Time

- In linear time logics we look at the execution paths individually
- In branching time logics we view the computation as a tree
 - computation tree: unroll the transition relation

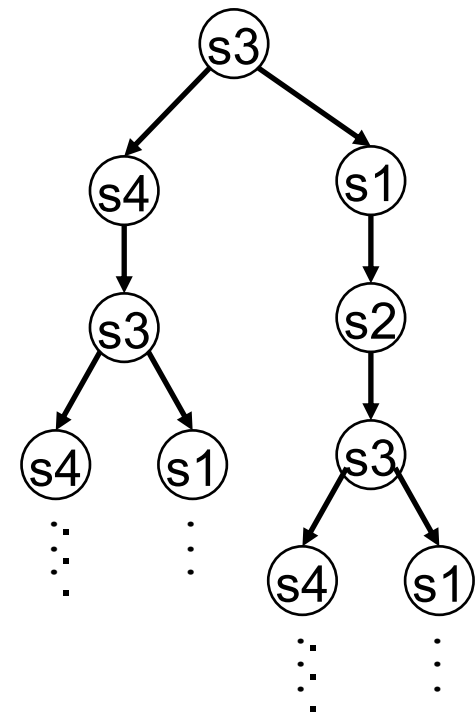
Transition System



Execution Paths



Computation Tree



Computation Tree Logic (CTL)

- In CTL we quantify over the paths in the computation tree
- We use the same four temporal operators: X, G, F, U
- However we attach path quantifiers to these temporal operators:
 - A : for all paths
 - E : there exists a path
- We end up with eight temporal operators:
 - AX, EX, AG, EG, AF, EF, AU, EU

CTL Semantics

Given a state s and CTL properties p and q

$s \models p$ iff $L(s, p) = \text{True}$, where $p \in AP$

$s \models \neg p$ iff not $s \models p$

$s \models p \wedge q$ iff $s \models p$ and $s \models q$

$s \models p \vee q$ iff $s \models p$ or $s \models q$

$s_0 \models EX p$ iff there exists a path s_0, s_1, s_2, \dots such that
 $s_1 \models p$

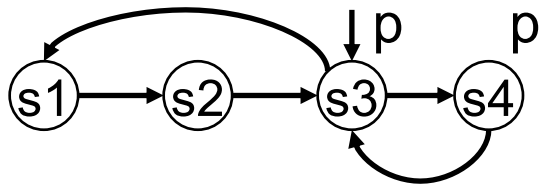
$s_0 \models AX p$ iff for all paths s_0, s_1, s_2, \dots , $s_1 \models p$

CTL Semantics

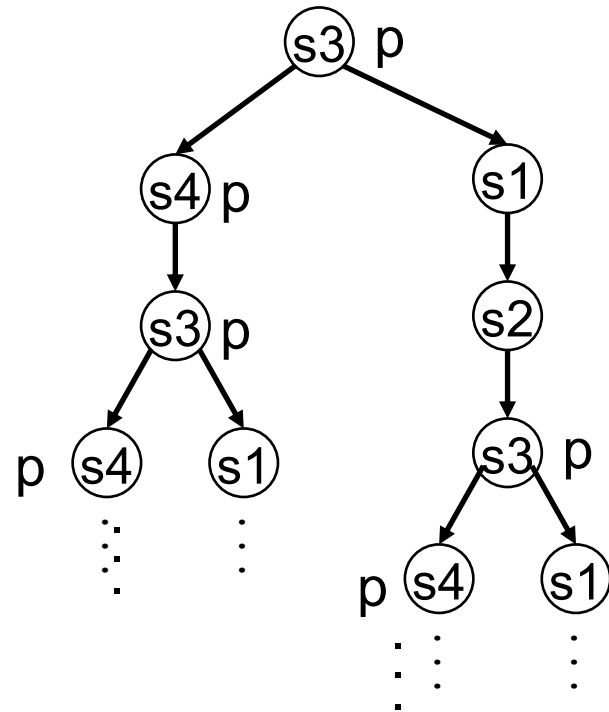
- $s_0 \models EG p$ iff there exists a path s_0, s_1, s_2, \dots such that for all $i \geq 0$, $s_i \models p$
- $s_0 \models AG p$ iff for all paths s_0, s_1, s_2, \dots , for all $i \geq 0$, $s_i \models p$
- $s_0 \models EF p$ iff there exists a path s_0, s_1, s_2, \dots such that there exists an $i \geq 0$ such that $s_i \models p$
- $s_0 \models AF p$ iff for all paths s_0, s_1, s_2, \dots , there exists an $i \geq 0$, such that, $s_i \models p$
- $s_0 \models p EU q$ iff there exists a path s_0, s_1, s_2, \dots , such that, there exists an $i \geq 0$ such that $s_i \models q$ and for all $0 \leq j < i$, $s_j \models p$
- $s_0 \models p AU q$ iff for all paths s_0, s_1, s_2, \dots , there exists an $i \geq 0$ such that $s_i \models q$ and for all $0 \leq j < i$, $s_j \models p$

CTL Properties

Transition System



Computation Tree



$s3 \models p$
 $s4 \models p$
 $s1 \models \neg p$
 $s2 \models \neg p$

$s3 \models EX p$
 $s3 \models EX \neg p$
 $s3 \models \neg AX p$
 $s3 \models \neg AX \neg p$
 $s3 \models EG p$
 $s3 \models \neg EG \neg p$
 $s3 \models AF p$
 $s3 \models EF \neg p$
 $s3 \models \neg AF \neg p$

CTL Equivalences

- CTL basis: EX, EU, EG

$$AX p = \neg EX \neg p$$

$$AG p = \neg EF \neg p$$

$$AF p = \neg EG \neg p$$

$$p AU q = \neg((\neg q EU (\neg p \wedge \neg q)) \vee EG \neg q)$$

$$EF p = \text{True} EU p$$

- Another CTL basis: EX, EU, AU

CTL Model Checking

- Given a transition system $T = (S, I, R)$ and a CTL property p
 $T \models p$ iff for all initial state $s \in I$, $s \models p$

Model checking problem: Given a transition system T and a CTL property p , determine if T is a model for p (i.e., if $T \models p$)

For example:

$T \models? AG(\neg(pc1=c \wedge pc2=c))$

$T \models? AG(pc1=w \Rightarrow AF(pc1=c)) \wedge AG(pc2=w \Rightarrow AF(pc2=c))$

CTL vs. LTL

- Leslie Lamport: "Sometime" is Sometimes "Not Never" - On the Temporal Logic of Programs. POPL 1980: 174-185
- E. Allen Emerson, Joseph Y. Halpern: "Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic. J. ACM 33(1): 151-178 (1986)
- Question: Are CTL and LTL equivalent?

CTL vs. LTL

- CTL and LTL are not equivalent
 - There are properties that can be expressed in LTL but cannot be expressed in CTL
 - For example: $FG\ p$
 - There are properties that can be expressed in CTL but cannot be expressed in LTL
 - For example: $AG(EF\ p)$
- Hence, expressive power of CTL and LTL are not comparable

CTL*

- CTL* is a temporal logic which is strictly more powerful than CTL and LTL
- CTL* also uses the temporal operators X, F, G, U and the path quantifiers A and E, but temporal operators can also be used without path quantifiers

CTL*

- CTL and CTL* correspondence
 - Since and CTL property is also a CTL* property, CTL* is clearly as expressive as CTL
- Any LTL f property corresponds to the CTL* property $A f$
 - i.e., LTL properties have an implicit “for all paths” quantifier in front of them
 - Note that, according to our definition, an LTL property f holds for a transition system T , if and only if, for all execution paths of T , f holds
 - So, LTL property f holds for the transition system T if and only if the CTL* property $A f$ holds for all initial states of T

CTL*

- CTL* is more expressive than CTL and LTL
- Following CTL* property cannot be expressed in CTL or LTL
 - $\neg A(\text{FG } p) \vee \text{AG}(\text{EF } p)$

Model Checking Algorithm for Finite State Systems

[Clarke and Emerson 81], [Queille and Sifakis 82]

CTL Model checking problem: Given a transition system $T = (S, I, R)$, and a CTL formula f , does the transition system satisfy the property?

CTL model checking problem can be solved in

$$O(|f| \times (|S| + |R|))$$

Note that the complexity is linear in the size of the formula and the transition system

- Recall that the size of the transition system is exponential in the number of variables and concurrent components (this is called the ***state space explosion*** problem)

CTL Model Checking Algorithm

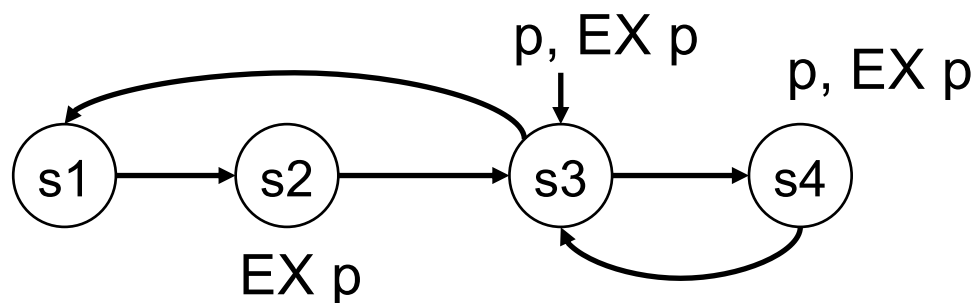
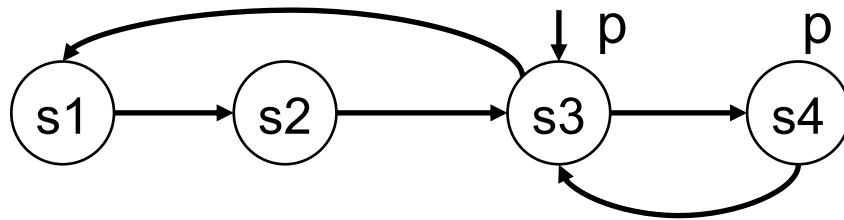
- Translate the formula to a formula which uses the basis
 - EX p, EG p, p EU q
- Start from the innermost subformulas
 - ***Label the states in the transition system with the subformulas that hold in that state***
 - Initially states are labeled with atomic properties
- Each (temporal or boolean) operator has to be processed once
- Processing of each operator takes $O(|S|+|R|)$

CTL Model Checking Algorithm

- Boolean operators are easy
 - $\neg p$: Each state which is not labeled with p should be labeled with $\neg p$
 - $p \wedge q$: Each state which is labeled with both p and q should be labeled with $p \wedge q$
 - $p \vee q$: Each state which is labeled with p or q should be labeled with $p \vee q$

CTL Model Checking Algorithm: EX p

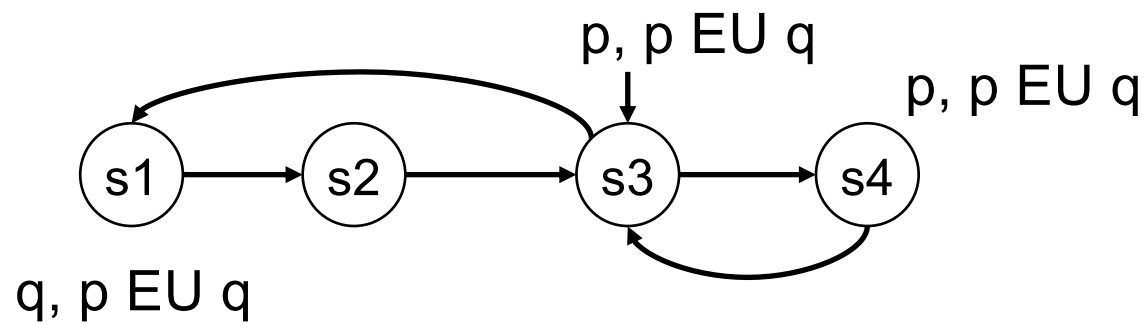
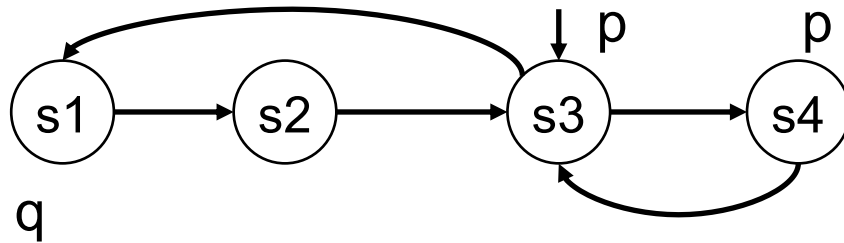
- EX p is easy to do in $O(|S|+|R|)$
 - All the nodes which have a next state labeled with p should be labeled with EX p



CTL Model Checking Algorithm: $p \text{ EU } q$

- $p \text{ EU } q$: Find the states which are the source of a path where $p \text{ U } q$ holds
 - Find the nodes that reach a node that is labeled with q by a path where each node is labeled with p
 - Label such nodes with $p \text{ EU } q$
 - It is a reachability problem which can be solved in $O(|S|+|R|)$
 - First label the nodes which satisfy q with $p \text{ EU } q$
 - For each node labeled with $p \text{ EU } q$, label all its previous states that are labeled with p with $p \text{ EU } q$

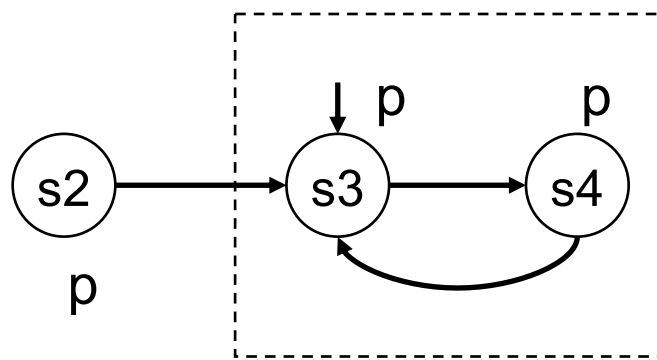
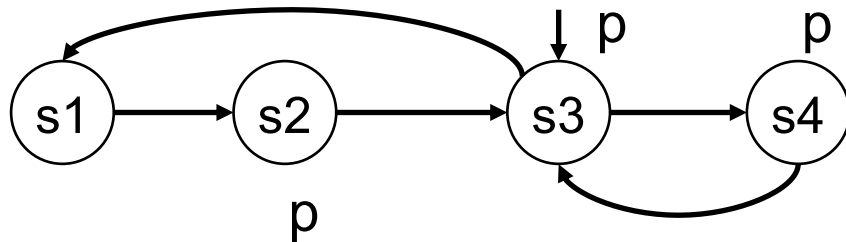
CTL Model Checking Algorithm: $p \text{ EU } q$



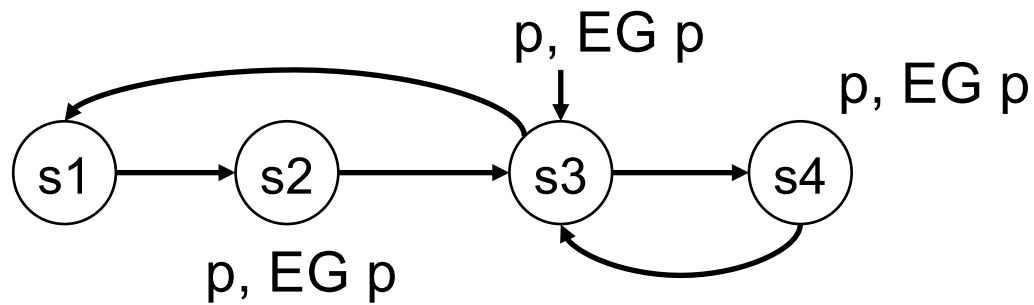
CTL Model Checking Algorithm: EG p

- EG p : Find infinite paths where each node on the path is labeled with p , and label nodes in such paths with EG p
 - First remove all the states which do not satisfy p from the transition graph
 - Compute the strongly connected components of the remaining graph, and then find the nodes which can reach the strongly connected components (both of which can be done in $O(|S|+|R|)$)
 - Label the nodes in the strongly connected components and the nodes that can reach the strongly connected components with EG p

CTL Model Checking Algorithm: EG p



A strongly connected component



Verification vs. Falsification

- Verification:
 - Show: initial states \subseteq truth set of p
- Falsification:
 - Find: a state \in initial states \cap truth set of $\neg p$
 - Generate a counter-example starting from that state
- Model checking algorithms can be modified to generate a counter-example paths if the property is not satisfied
 - without increasing the complexity
- The ability to find counter-examples is one of the biggest strengths of the model checkers

Counter-Example Generation

- Remember: Given a transition system $T = (S, I, R)$ and a CTL property p $T \models p$ iff for all initial state $s \in I$, $s \models p$
- Verification vs. Falsification
 - Verification:
 - Show: initial states \subseteq truth set of p
 - Falsification:
 - Find: a state \in initial states \cap truth set of $\neg p$
 - Generate a counter-example starting from that state
- The ability to find counter-examples is one of the biggest strengths of the model checkers

General Idea

- We can define two temporal logics using subsets of CTL operators
 - ACTL: CTL formulas which only use the temporal operators AX, AG, AF and AU and all the negations appear only in atomic properties (there are no negations outside of temporal operators)
 - ECTL: CTL formulas which only use the temporal operators EX, EG, EF and EU and all the negations appear only in atomic properties
- Given an ACTL property its negation is an ECTL property

An Example

- If we wish to check the property $AG(p)$
- We can use the equivalence:
 $AG(p) \equiv \neg EF(\neg p)$

If we can find an initial state which satisfies $EF(\neg p)$, then we know that the transition system T , does not satisfy the property $AG(p)$

Another Example

- If we wish to check the property $AF(p)$
- We can use the equivalence:
$$AF(p) \equiv \neg EG(\neg p)$$

If we can find an initial state which satisfies $EG(\neg p)$, then we know that the transition system T , does not satisfy the property $AF(p)$

Counter-Example Generation for ACTL

- Given an ACTL property p , we negate it and compute the set of states which satisfy its negation $\neg p$
 - $\neg p$ is an ECTL property
- If we can find an initial state which satisfies $\neg p$ then we generate a counter-example path for p starting from that initial state by following the states that are marked with $\neg p$
 - Such a path is called a *witness* for the ECTL property $\neg p$

Counter-example generation for ACTL

- In general the counter-example for an ACTL property (equivalently a witness to an ECTL property) is not a single path
- For example, the counter example for the property $AF(AGp)$ would be a witness for the property $EG(EF\neg p)$
 - It is not possible to characterize the witness for $EG(EF\neg p)$ as a single path
- However it is possible to generate tree-like transition graphs containing counter-example behaviors as a counter-example:
 - Edmund M. Clarke, Somesh Jha, Yuan Lu, Helmut Veith: “Tree-Like Counterexamples in Model Checking”. LICS 2002: 19-29

Counter-example generation for LTL

- Recall that, an LTL property f holds for a transition system T , if and only if, for all execution paths of T , f holds
- Then, to generate a counter-example for an LTL property f , we need to show that there exists an execution path for which $\neg f$ holds.
 - Given an LTL property f , a counter-example is an execution path for which $\neg f$ holds

What About LTL and CTL* Model Checking?

- The complexity of the model checking problem for LTL and CTL* are:
 - $(|S|+|R|) \times 2^{O(|f|)}$
- Typically the size of the formula is much smaller than the size of the transition system
 - So the exponential complexity in the size of the formula is not very significant in practice