

# Temporal Logics

Abhik Roychoudhury  
CS 5219  
School of Computing

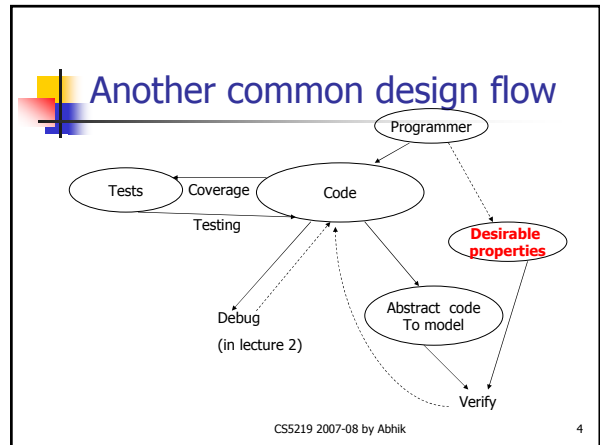
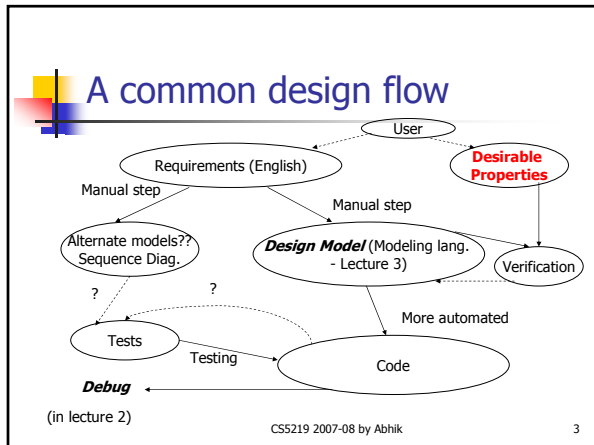
CS5219 2007-08 by Abhik 1

# Turing award

- 1996 – Amir Pnueli
- Weizmann Institute of Science (1996)  
**Citation**  
"For seminal work introducing temporal logic into computing science and for outstanding contributions to program and system verification."

2007 Turing award for Model checking – the use of temporal logics in Validation.

CS5219 2007-08 by Abhik 2



# Background

- Checking of programs
  - $P \models \varphi$ 
    - How to represent P ?
      - Implementation
    - How to represent  $\varphi$  ?
      - Specification
- Today we will answer the second question

CS5219 2007-08 by Abhik 5

# What about the first question?

- Assume a transition system like model of the program
  - $M = (I, S, \rightarrow)$ 
    - I is set of initial states
    - S is set of states
    - $\rightarrow$  is set of transitions
- What are the states and transitions in a program?

CS5219 2007-08 by Abhik 6

## States and Transitions

- States of a sequential program
  - (PC, Value of x, Value of y, ...)
  - Infinitely many states in general
- Transition of a program
  - $s1 \rightarrow s2$
  - $s1, s2$  are states as defined above
  - Move from  $s1$  to  $s2$  accomplished by executing a statement

CS5219 2007-08 by Abhik

7

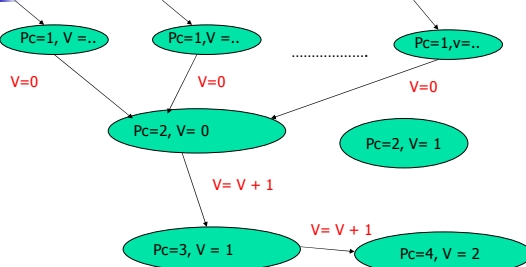
## Exercise 0

- Draw the transition system corresponding to the following program
  - $V = 0;$
  - $V = V + 1;$
  - $V = V + 1;$

CS5219 2007-08 by Abhik

8

## Solution



CS5219 2007-08 by Abhik

9

## Exercise 1

- What changes will we need to make in defining states and transitions of say a multi-threaded Java program?

CS5219 2007-08 by Abhik

10

## Exercise 2

- Our definition of states and transitions results in infinitely many states even for sequential programs. What abstractions can you suggest?
  - Should we abstract the PC?
  - Should we abstract the variable values?

CS5219 2007-08 by Abhik

11

## Exercise 3

- Revisit the program
  - $V = 0;$
  - $V = V + 1;$
  - $V = V + 1;$
- Abstract  $V$  using the propositions
  - $\{V == 0\}$
- Reconstruct the transition system.

CS5219 2007-08 by Abhik

12

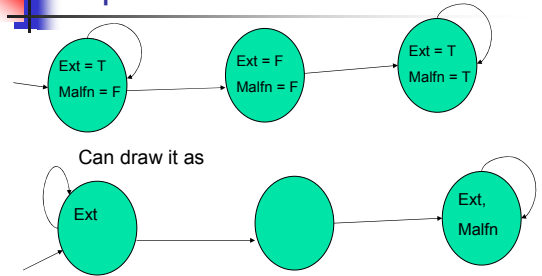
## Program Models

- Kripke Structure
  - Transition System (States and Transitions)
    - Can depict the control flow in a program.
  - Set of Propositions Prop
    - Abstraction of the data variables.
  - Truth/Falsehood of each proposition in Prop in each state of the transition system.
    - Abstraction of data flow in the program.

CS5219 2007-08 by Abhik

13

## Kripke Structure



CS5219 2007-08 by Abhik

14

## Kripke Structure

- $M = (S, I, \rightarrow, L)$ 
  - $S$ , set of states
  - $I \subseteq S$ , set of initial states
  - $\rightarrow$ , Transition Relation
  - $L$ , Labeling function
    - Label states with propositions – these are the propositions which are true in the state.

CS5219 2007-08 by Abhik

15

## Model Checking – Usage (I)

- Model the software to be constructed.
  - Say in Promela
- Specify the properties to be verified.
  - Temporal logics --- today !
- Automated verification
  - SPIN model checker.
  - Does not verify the software, but checks the software design.

CS5219 2007-08 by Abhik

16

## Model Checking – Usage (II)

- Generate models (Kripke Structure) from the software
  - Represent the control flow explicitly (Control Flow Graph or variants).
  - Abstract data store via predicate abstraction.
    - Implicitly blows up the CFG.
  - State space search now involves traversing the blown up graph (note: symbolic search).

CS5219 2007-08 by Abhik

17

## Our Program Checking

- Specify property in temporal logic (Today's lecture)
  - Clock time is not explicitly represented.
  - Temporal modalities to capture dynamic program behavior
- Verify property automatically via search
  - Return "yes" if true
  - Return counterexample "evidence" if false.

CS5219 2007-08 by Abhik

18

## Organization

- Basics
- Linear Time Temporal Logics
- Branching Time Logics
- Difference between Formula Checking and Satisfiability
- Optional material
  - Fixed point characterizations

CS5219 2007-08 by Abhik

19

## Why study new logics?

- Propositional/first-order logics can only express properties of states, not properties of how the states evolve.
- We study behaviors by looking at all execution traces of the system.
- Our properties refer to dynamic system behaviors
  - **Eventually**, the system reaches a stable state
  - **Never** a deadlock can occur
- We want to maintain more than input-output relationships (typical for transformational systems).

CS5219 2007-08 by Abhik

20

## Temporal Logic

- Propositional / First-order logic formulae
  - Capture properties of states
  - Cannot capture properties of state changes by default.
- Temporal Logic formulae
  - Capture properties of evolution of states (program behavior)
  - Possible program behaviors can be
    - Execution traces OR Execution Tree

CS5219 2007-08 by Abhik

21

## What is temporal ?

- Consider ordering of events in system execution.
  - Describes properties of such orderings
    - Whenever  $V = 0$ ,  $U$  is not 0
    - Always  $V = 0$
    - Whenever  $V=0$ , eventually  $U = 0$
  - Exact time is not represented e.g.
    - $V = 0$  will happen at  $t = 22$  secs.

CS5219 2007-08 by Abhik

22

## Purpose of TL

- Describe properties of program behavior
- What are the units of behavior
  - Computation Tree of the program
  - Set of computation traces of the program
- Linear and branching time logics.
- TL can specify properties about behavior of
  - Terminating and non-terminating programs.
  - Sequential as well as concurrent programs.

CS5219 2007-08 by Abhik

23

## LTL: What ?

- Linear-time Temporal Logic
- An LTL formula is interpreted over infinite execution traces.
- LTL can capture properties of
  - Usual terminating programs
  - Non-terminating reactive programs which are continuously interacting with environment
    - Example: Controller software

CS5219 2007-08 by Abhik

24

## LTL syntax

- An LTL property  $\varphi$  is true of a program model iff **all** its traces satisfy  $\varphi$ 
  - Why should be there be more than one trace?**
- $\varphi = X\varphi \mid G\varphi \mid F\varphi \mid \varphi U \varphi \mid \varphi R \varphi \mid \neg\varphi \mid \varphi \wedge \varphi \mid \text{Prop}$
- Prop denotes the set of Propositions.
- X, G, F, U, R are temporal operators.
- Building a temporal logic above propositional logic (included above)

CS5219 2007-08 by Abhik

25

## LTL semantics

- $M, \pi \models \varphi$ 
  - Path  $\pi = s_0, s_1, s_2, \dots$  in model M satisfies property  $\varphi$
- $M, \pi^k \models \varphi$ 
  - Notation for Path  $s_k, s_{k+1}, \dots$  in model M satisfies the property  $\varphi$
- Semantics for different temporal operators are now given.

CS5219 2007-08 by Abhik

26

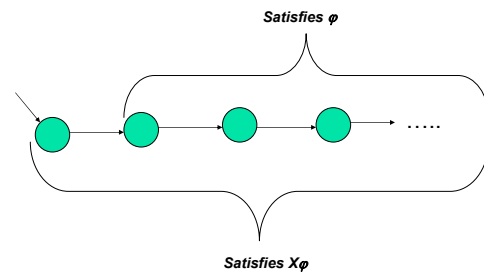
## Semantics of LTL operators

- $M, \pi \models X\varphi$  iff  $M, \pi^1 \models \varphi$ 
  - Path starting from **next state** satisfies  $\varphi$
- $M, \pi \models F\varphi$  iff  $\exists k \geq 0 M, \pi^k \models \varphi$ 
  - Path starting from an **eventually** reached state satisfies  $\varphi$
- $M, \pi \models G\varphi$  iff  $\forall k \geq 0 M, \pi^k \models \varphi$ 
  - Path **always** satisfies  $\varphi$  (all suffixes of the path satisfy  $\varphi$ )

CS5219 2007-08 by Abhik

27

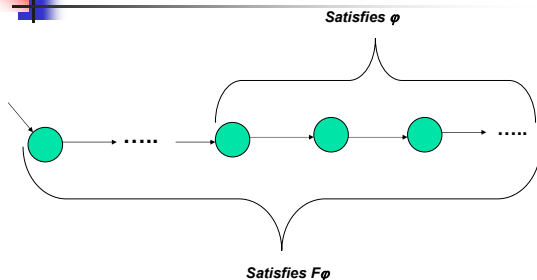
## next-state operator in LTL



CS5219 2007-08 by Abhik

28

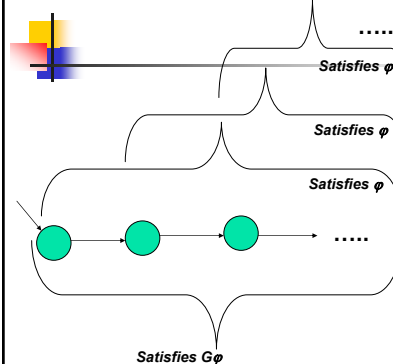
## Finally operator in LTL



CS5219 2007-08 by Abhik

29

## Globally operator in LTL



CS5219 2007-08 by Abhik

30

### Semantics of LTL operators

- $M, \pi \models \varphi_1 U \varphi_2$  iff  $\exists k \geq 0$  such that
  - $M, \pi^k \models \varphi_2$ , and
  - $\forall 0 \leq j < k, M, \pi^j \models \varphi_1$

A trace satisfying  $p U q$ , where  $p, q \in \text{Prop}$

CS5219 2007-08 by Abhik 31

### Until operator in LTL

CS5219 2007-08 by Abhik 32

### Semantics of LTL operators

- $M, \pi \models \varphi_1 R \varphi_2$  iff
  - Either  $\forall k \geq 0, M, \pi^k \models \varphi_2$
  - OR both of the following hold
    - $\exists k \geq 0, M, \pi^k \models \varphi_1$
    - $\forall 0 \leq j \leq k, M, \pi^j \models \varphi_2$
- $\varphi_1$  releases the req. for  $\varphi_2$  to hold.

CS5219 2007-08 by Abhik 33

### Release operator in LTL – (1)

CS5219 2007-08 by Abhik 34

### Release operator in LTL – (2)

CS5219 2007-08 by Abhik 35

### Example 1

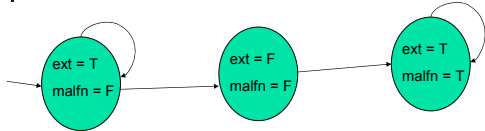
Property:  $G \text{ ext}$

$M \models G \text{ ext}$  iff for traces  $\pi$  in  $M$  we have  $M, \pi \models G \text{ ext}$

Any trace which does not satisfy  $G \text{ ext}$  is a counter-example. How many counterexamples can you find in the above ??

CS5219 2007-08 by Abhik 36

## Example 2



What about the formulae: **FG ext**, **GF ext** ??

How many counter-examples for each ?

CS5219 2007-08 by Abhik

37

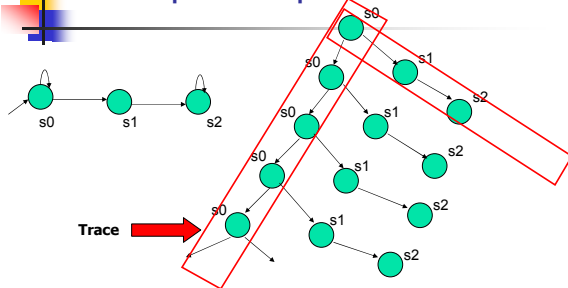
## Computation traces/trees

- LTL describes properties of computation traces (a property holds in a sys. if all traces satisfy it).
- Alternate way to characterize system dynamics
  - Computation tree
  - Start from an initial state and unfold the Kripke structure to construct an infinite tree (in general).
- Logics to describe properties of such trees
  - Existential / Universal quantification over paths
  - Temporal operators to specify properties of a path.

CS5219 2007-08 by Abhik

38

## Example computation tree



CS5219 2007-08 by Abhik

39

## A logic for trees

- $s = \text{Prop} \mid \neg s \mid s \wedge s \mid \mathbf{A} p \mid \mathbf{E} p$
- $p = \mathbf{X} s \mid \mathbf{G} s \mid \mathbf{F} s \mid \mathbf{U} s \mid \mathbf{R} s$
- $p$  denotes formulae of paths.
- $s$  denotes formulae of states.
- The temporal operators are as before.
- Computation Tree logic (CTL).
  - All state formulae as defined above.

CS5219 2007-08 by Abhik

40

## CTL semantics

- A model satisfies a CTL formula iff its **initial states** satisfy the formula.
- A state  $x$  satisfies the formula  $\mathbf{A} p$  iff all outgoing paths from  $x$  satisfy the formula  $p$ 
  - Note that  $p$  must be a path formula
- A state  $x$  satisfies the formula  $\mathbf{E} p$  iff there exists an outgoing path from  $x$  satisfying the formula  $p$
- What about the temporal operators ?

CS5219 2007-08 by Abhik

41

## CTL semantics

- Semantics of temporal operators (X,F,G,U,R)
  - We learnt this before !
  - Try a path proof obligation  $\pi \models \mathbf{FEG} p$
- Exercise : Do it Now !
  - Meaning of AG, EG, AF, EF (intuitively)
  - Duality of (R, U), (F, G), (A, E)
  - Express F in terms of U
  - Minimal set of temporal & path operators

CS5219 2007-08 by Abhik

42

## Exercise

```

    graph LR
      s0((s0)) --> s0
      s0 --> s1((s1))
      s1 --> s2((s2))
      s2 --> s2
  
```

Suppose we want to check  
 $M \models \text{AGEF } p$   
 Show all the state and path proof obligations that will be encountered by following through the semantic rules of Computation Tree Logic (CTL).

**Model M**

CS5219 2007-08 by Abhik 43

## CTL formulae

- AG f (invariant property)
  - $M \models \text{AG } f$  (CTL formula) if and only if M satisfies the LTL formula G f
- AG (f  $\Rightarrow$  EF g)
  - what does it mean ?
  - Involves both path quantifiers
  - Not expressible in LTL
- Does that mean CTL is strictly more powerful than LTL ?

CS5219 2007-08 by Abhik 44

## CTL and LTL

The simple idea of inserting A operators will not work.

```

    graph LR
      p1((p)) --> p1
      p1 --> np1((-p))
      np1 --> p2((p))
      p2 --> np2((-p))
      np2 --> p1
  
```

Satisfies the LTL formula FG p  
 What about the CTL formula AFAG p ?

The issues of Temporal Logic expressivity are much deeper and outside the scope of this course.

CS5219 2007-08 by Abhik 45

## Expressivity

CS5219 2007-08 by Abhik 46

## CTL\*

- $s = \text{Prop} \mid \neg s \mid s \wedge s \mid A p \mid E p$
- $p = s \mid \neg p \mid p \wedge p \mid X p \mid F p \mid G p \mid p U p \mid p R p$ 
  - s denotes formulae interpreted over states
  - p denotes formulae interpreted over paths
  - CTL\* is the set of all state formulae above
  - A p is a state formula not expressed in prop. Logic
- Semantics of path formulae
  - Minimally modified to handle ...
  - A path satisfies a state formula iff its first state satisfies the formula.

CS5219 2007-08 by Abhik 47

## CTL and CTL\*

- CTL is a sublogic of CTL\*
  - Temporal operators in CTL\*: X, F, G, U, R
  - Path Quantifiers: A, E
  - CTL enforces the occurrence of a temporal operator to be immediately preceded by a path quantifier.
  - AGF $\phi$  is not allowed.

CS5219 2007-08 by Abhik 48



## Some common CTL formulae

- **AG p**
  - **Invariant: always p.**
- **EF p**
  - **Reachability: of a state where p holds.**
- **AF p**
  - **Inevitability of reaching a state where p holds.**
- **AG EF p**
  - **Recovery: from any state we can reach a state where p holds.**
- **AG (p ⇒ AF q)**
  - **Non-starvation : p request is always provided q response.**

CS5219 2007-08 by Abhik

49

## Exercise (1)

Consider a resource allocation protocol where  $n$  processes  $P_1, \dots, P_n$  are contending for exclusive access of a shared resource. Access to the shared resource is controlled by an arbiter process. The atomic proposition  $req_i$  is true only when  $P_i$  explicitly sends an access request to the arbiter. The atomic proposition  $gnt_i$  is true only when the arbiter grants access to  $P_i$ . Now suppose that the following LTL formula holds for our resource allocation protocol.

- **G (req<sub>i</sub> ⇒ (req<sub>i</sub> U gnt<sub>i</sub> ) )**

CS5219 2007-08 by Abhik

50

## Exercises (1)

- Explain in English what the property means.
- Is this a desirable property of the protocol ?
- Suppose that the resource allocation protocol has a distributed implementation so that each process is implemented in a different site. Does the LTL property affect the communication overheads among the processes in any way ?

CS5219 2007-08 by Abhik

51

## Exercises (2)

- Express each of the following properties (stated in English) as an LTL formula. Assume that  $p$ ,  $q$  and  $r$  are atomic propositions.
  - Always if  $p$  occurs, then eventually  $q$  occurs followed immediately by  $r$ .
  - Any occurrence of  $p$  is followed eventually by an occurrence of  $q$ . Furthermore,  $r$  never occurs between  $p$  and  $q$ .

CS5219 2007-08 by Abhik

52

## Exercises (3)

- Consider the LTL formula  $GFp$  and the CTL formula  $AGEFp$  where  $p$  is an atomic proposition. Give an example of a Kripke Structure which satisfies  $AGEFp$  but does not satisfy  $GFp$ . You may assume that  $p$  is the only atomic proposition for constructing the labeling function.
- Are the following LTL formulae equivalent
  - $G(p \Rightarrow X p)$
  - $G(p \Rightarrow G p)$

CS5219 2007-08 by Abhik

53

## Exercises (4)

Assume that  $p$  is an atomic proposition. What can you say about the equivalence of the following pairs of temporal formulae? If they are equivalent, then provide a formal proof. If not construct an example Kripke Structure to show that they are not equivalent.

- (a) the LTL formula  $GFp$  and the CTL formula  $AGFp$
- (b) the CTL formulae  $AGAFp$  and the CTL formula  $AGEFp$
- (c) the LTL formula  $GFp$  and the CTL formula  $AGAFp$

CS5219 2007-08 by Abhik

54

## Satisfaction

- A CTL formula is satisfiable if some state of some Kripke structure satisfies it.
  - Otherwise unsatisfiable. Examples ??
  - Similarly for LTL formula .
- A CTL formula is valid if all states of all Kripke structures satisfy it.

CS5219 2007-08 by Abhik

55

## Formula Equivalence

- Two CTL properties are equivalent iff they are satisfied by exactly the same states of any Kripke structure.
  - $EF p$  and  $E(\text{true} \cup p)$
- Where does model checking stand ??
  - Is it checking for satisfiability of a temporal property ? Is it checking for validity ?

CS5219 2007-08 by Abhik

56

## Model Checking

- ... is not checking for satisfiability / validity.
- It is checking for satisfaction of a temporal property for a **given** Kripke structure.
  - This is a very different problem from traditional satisfiability checking !!
- We will discuss MC in next class.
  - But some warm up for now !
  - This is the more formal part of our discussion. Use it more for your own understanding of TL. The material is optional for exam purposes.

CS5219 2007-08 by Abhik

57

## Temporal Logics- Optional Material (only for self-study)

Abhik Roychoudhury  
National University of Singapore

CS5219 2007-08 by Abhik

58

## Fixed point characterizations

- An alternate semantic understanding of temporal formulae such as CTL properties.
- Yields a procedure for model checking these properties directly
  - Correct by construction model checking algorithm.

CS5219 2007-08 by Abhik

59

## Intuition -- (1)

- Give semantics of CTL formulae by associating a formula with the states in a Kripke structure in which the formula will be true.
- A set of states drawn from a Kripke Structure forms a predicate.
- Define functions on sets of states
  - $F: 2^S \rightarrow 2^S$
  - **S is the set of all states drawn from Kripke structure**
  - **Such functions are called predicate transformers.**

CS5219 2007-08 by Abhik

60

## Intuition – (2)

- Define a CTL formula as the set of states obtained by
  - Repeated application of a predicate transformer
  - Starting from null set or set of all states.
  - Ending when one more application of the transformer does not change the result
    - Fixed point is reached.

CS5219 2007-08 by Abhik

61

## Set of States

- $M = (S, I, \rightarrow, L)$ 
  - Assume  $S$  is finite
  - $\rightarrow$  (transition relation)
  - $L$  (Labeling function)
- $x \in 2^S$ 
  - $x$  is a predicate over  $S$
- Powerset  $2^S$  comes with a natural partial order
  - Set inclusion

CS5219 2007-08 by Abhik

62

## Monotone Functions

- A function  $f : 2^S \rightarrow 2^S$ 
  - Monotonic:  $X \subseteq Y \Rightarrow f(X) \subseteq f(Y)$
  - Fixed point:  $f(X) = X$
  - $X, Y$  are subsets of  $S$  (set of states)
- Since it is transforming sets of states to another set of states, also called
  - Predicate Transformer

CS5219 2007-08 by Abhik

63

## Example

- $S = \{s0, s1\}$
- $f : 2^S \rightarrow 2^S$ 
  - $f(X) = X \cup \{s0\}$
  - Show that  $f$  is monotonic.
  - What are the fixed points of  $f$  ?
- $f : 2^S \rightarrow 2^S$ 
  - Exercise: Give an example of non-monotone function.

CS5219 2007-08 by Abhik

64

## Role of Monotone functions

- Always have least/greatest fixed points
- Meaning of CTL operators cast as lfp/gfp of monotone func. over  $2^S$ 
  - AG, EG, AF, EF, AU, EU, ...
- The fixed points can be computed
  - Straightforward checking procedure
    - Compute fixed-point and check for initial states within the fixed-point.

CS5219 2007-08 by Abhik

65

## Why fixed points are important ?

- Meaning of CTL operators can be expressed as lfp, gfp of monotone functions.
  - EG, EU, AG, AF, ...
- These lfp, gfp can be easily computed.
- Leads to a straightforward model checking algorithm for CTL formulae.

CS5219 2007-08 by Abhik

66

## Fixed points

- If  $S$  is finite, then for a monotone function  $f : 2^S \rightarrow 2^S$ 
  - $\exists I \in \text{nat } \text{lfp } f = f^I(\emptyset)$
  - $\exists I \in \text{nat } \text{gfp } f = f^I(S)$
- Class Exercise:
  - Let us prove this theorem now.

CS5219 2007-08 by Abhik

67

## LFP computation procedure

- Function  $\text{lfp } (f : \text{PredicateTransformer}) : \text{Predicate}$
- Begin
- $Q := \emptyset$ ; // Null-set
- $Q1 := f(Q)$ ;
- while  $(Q1 \neq Q)$  do
- $Q := Q1$ ;  $Q1 := f(Q)$
- endwhile;
- return( $Q$ )
- End.

CS5219 2007-08 by Abhik

68

## GFP computation procedure

- Function  $\text{gfp } (f : \text{PredicateTransformer}) : \text{Predicate}$
- Begin
- $Q := S$ ; // All states in the Kripke Structure
- $Q1 := f(Q)$ ;
- while  $(Q1 \neq Q)$  do
- $Q := Q1$ ;  $Q1 := f(Q)$
- endwhile;
- return( $Q$ )
- End.

CS5219 2007-08 by Abhik

69

## Defining CTL operators

- $M = (S, I, \rightarrow, L)$  is a finite Kripke structure
- $\varphi$  is a CTL formula
- $[\varphi] \subseteq S$  denotes the set of states satisfying  $\varphi$
- Say we define a predicate transformer
  - $f_\varphi(Y) = [\varphi] \cap \{s \mid \exists s' \ s \rightarrow s' \text{ and } s' \in Y\}$

CS5219 2007-08 by Abhik

70

## Defining EG

- $\{s \mid \exists s' \ s \rightarrow s' \text{ and } s' \in Y\}$ 
  - Stands for  $[\text{EX } Y]$ , the set of states in  $M$  which satisfy  $\text{EX } Y$ .
  - For convenience we will avoid the [...]
- $f_\varphi(Y) = \varphi \cap \text{EX } Y$ 
  - Show that this transformer is monotonic.
  - You will need the definition of  $\text{EX}$

CS5219 2007-08 by Abhik

71

## Defining EG

- Show that  $\text{EG}\varphi$  is a fixed point of
  - $f_\varphi(Y) = \varphi \cap \text{EX } Y$ .
  - Any state satisfying  $\text{EG}\varphi$  satisfies  $\varphi$  and there is an outgoing state satisfying  $\text{EG}\varphi$
  - The other direction of the proof ...
- Show that  $\text{EG}\varphi$  is the gfp of
  - $f_\varphi(Y) = \varphi \cap \text{EX } Y$
  - Any state in a fixed point of  $f_\varphi$  satisfies  $\text{EG } \varphi$ 
    - Complete the proof exploiting this intuition.

CS5219 2007-08 by Abhik

72

## Defining EU

- $E(\varphi \text{ U } \psi)$  is the least fixed point of
  - $f_{\varphi, \psi}(Y) = \psi \cup (\varphi \cap EX Y)$
- Cast the EU checking algo in terms of the LFP computation procedure outlined earlier
- Full discussion on CTL MC in next class.

CS5219 2007-08 by Abhik

73

## Property Equivalences

- $EG \varphi = \varphi \wedge EXEG\varphi$
- $E(\varphi \text{ U } \psi) = \psi \vee (\varphi \wedge EX E(\varphi \text{ U } \psi))$
- We can derive similar equivalences using the fixed point characterization of other CTL properties.

CS5219 2007-08 by Abhik

74

## Possible Readings

- Chapter 3 of "Model Checking"
  - Clarke, Grumberg, Peled
    - See IVLE E-reserves to access material
    - [QA76.76 Ver.C](#) RBR in Central Library
- Chapter 3 of Logic in Computer Science
  - (Huth and Ryan)
    - [QA76.9 Log.Hu](#) in Central Library
      - I also have a copy.
    - Chapter 3.9 contains discussion on fixed point characterizations.

CS5219 2007-08 by Abhik

75