NPTEL Phase-II
Video course on

Design Verification and Test of Digital VLSI Designs

Dr. Santosh Biswas
Dr. Jatindra Kumar Deka
IIT Guwahati

Module IV: Temporal Logic

Lecture I: Introduction to formal methods for design verification

Design Cycle

Specification
Design
Implementation
Testing
Installation/marketing
Maintenance

Design Cycle

Specification

Suppose we have to design the controller of a washing machine. There are certain aspects of washing clothes that the system has to take care of, like:

- The drier is activated after the wash not before it.
- Water is poured in before the detergent and it is drained before activating the drier.
- Cold water is to be used in soft wash where hot water in heavy wash, etc.

Design Cycle

Specification
Design
Implementation
Testing
Installation/marketing
Maintenance

Design Cycle

Specification
Design
Implementation
Testing
Installation/marketing
Maintenance

Bugs reported at early phase of design incur less cost for debugging.

Design Cycle

Specification
Design
Verification
Implementation
Testing
Installation/marketing
Maintenance

Verification is used to capture the bugs at the early phase of design cycle.

Simulation

Simulation:

- Exhaustive Simulation
- Non exhaustive Simulation

Simulation

Simulation:

- Exhaustive Simulation
- Non exhaustive Simulation

Number of test cases are exponential to the number of state variables.

Non Exhaustive Simulation

Instead of using all possible combinations, simulation is done for some selected input combinations.

Non Exhaustive Simulation

Instead of using all possible combinations, simulation is done for some selected input combinations.

To find the appropriate subset is a complex problem.

- Test case generation

Non Exhaustive Simulation

Instead of using all possible combinations, simulation is done for some selected input combinations.

To find the appropriate subset is a complex problem.

- Test case generation

We may not cover all possible error cases.

Non Exhaustive Simulation

Problems??

Non Exhaustive Simulation

Non Exhaustive Simulation: **Pentium Bug**

The intel pentium ™ processor for IBM compatible PC was first introduced into the market in May of 1993.

Non Exhaustive Simulation

Non Exhaustive Simulation: **Pentium Bug**

The intel pentium ™ processor for IBM compatible PC was first introduced into the market in May of 1993.

A year later an estimated two million had been sold, it was discovered that there was a flaw in the hardware of floating point division.

Non Exhaustive Simulation

Non Exhaustive Simulation: **Pentium Bug**

The intel pentium ™ processor for IBM compatible PC was first introduced into the market in May of 1993.

A year later an estimated two million had been sold, it was discovered that there was a flaw in the hardware of floating point division.

It uses the **SRT** floating point division.

(Sweeney, Robertson and Tocher)

Formal Verification

- Increased complexity of design

Formal Verification

- Increased complexity of design
- In formal verification, we deal with the abstract model of the system

Formal Verification

- Increased complexity of design
- In formal verification, we deal with the abstract model of the system
- Model helps us to build more complex systems

Formal Verification

- Increased complexity of design
- In formal verification, we deal with the abstract model of the system
- Model helps us to build more complex systems
- A model is easier to understand than a whole system

Formal verification

Construct a model (for the application) in which we can demonstrate that a certain property holds

Formal verification

Construct a model (for the application) in which we can demonstrate that a certain property holds

- System Model
- Specification (Property)

Formal verification

Construct a model (for the application) in which we can demonstrate that a certain property holds

Testing versus Formal Verification

What is wrong?

Why it is wrong?

Formal verification

- Approaches for formal verification
  - Propositional Logic
  - First Order Logic
  - Higher Order Logic

Deductive Verification Propositional logic

- Consisting of Boolean formulas comprising Boolean variables and connectives such as $\vee$ and $\wedge$.
- Gate-level logic networks can be described.
- Typical aim: checking if two models are equivalent (called **tautology checkers** or **equivalence checkers**).
- Since propositional logic is decidable, it is also decidable whether or not the two representations are equivalent.
- Tautology checkers can frequently cope with designs which are too large to allow simulation-based exhaustive validation.

First order logic (FOL)

- FOL includes quantification, using $\exists$ and $\forall$.
- Some automation for verifying FOL models is feasible.
- However, since FOL is undecidable in general, there may be cases of doubt.

Higher order logic (HOL)

- Higher Order Logic allows functions to be manipulated like other objects.

Higher order logic (HOL)

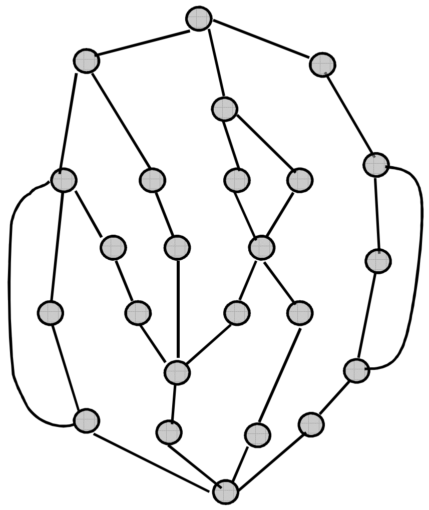
- Higher Order Logic allows functions to be manipulated like other objects.
- For higher order logic, proofs can hardly ever be automated and typically must be done manually with some proof-support.

Higher order logic (HOL)

- Higher Order Logic allows functions to be manipulated like other objects.
- For higher order logic, proofs can hardly ever be automated and typically must be done manually with some proof-support.
- Interactive theorem provers require a human user to give hints to the system.

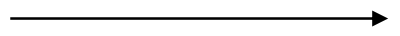
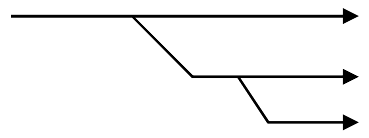
Temporal logic

- Logic extended with a notion of "time“
- Capture future behaviors



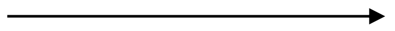
Temporal logic

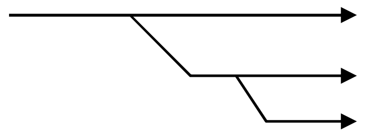
- Branching vs. linear time:
  - Linear time
    Models physical time
    At each time instant, only one of the future behaviors is considered.
  - Branching time (at each time instant, all possible future behaviors are considered).
    - Models different computational sequences of a system.
    - Nondeterministic selection of the path taken.

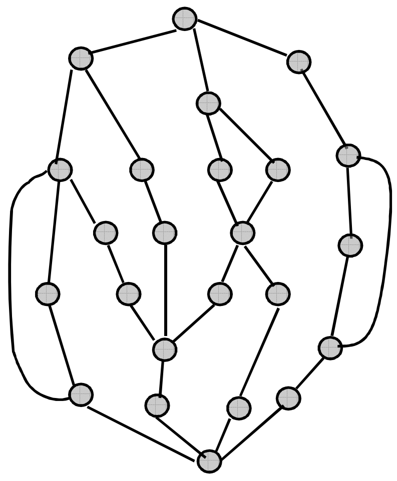



Temporal logic

- Branching vs. linear time:
  - Linear time
    Models physical time
    At each time instant, only one of the future behaviors is considered.
  - Branching time (at each time instant, all possible future behaviors are considered).
    - Models different computational sequences of a system.
    - Nondeterministic selection of the path taken.

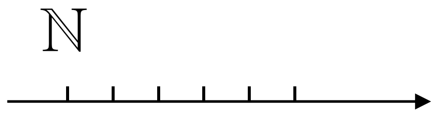


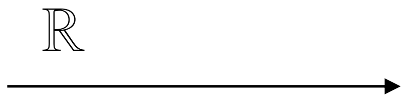




Temporal logic

- Discrete vs. continuous time
  - Discrete time
    Used by most temporal logics, mostly using natural numbers to model time.
  - Continuous time
    Using real numbers





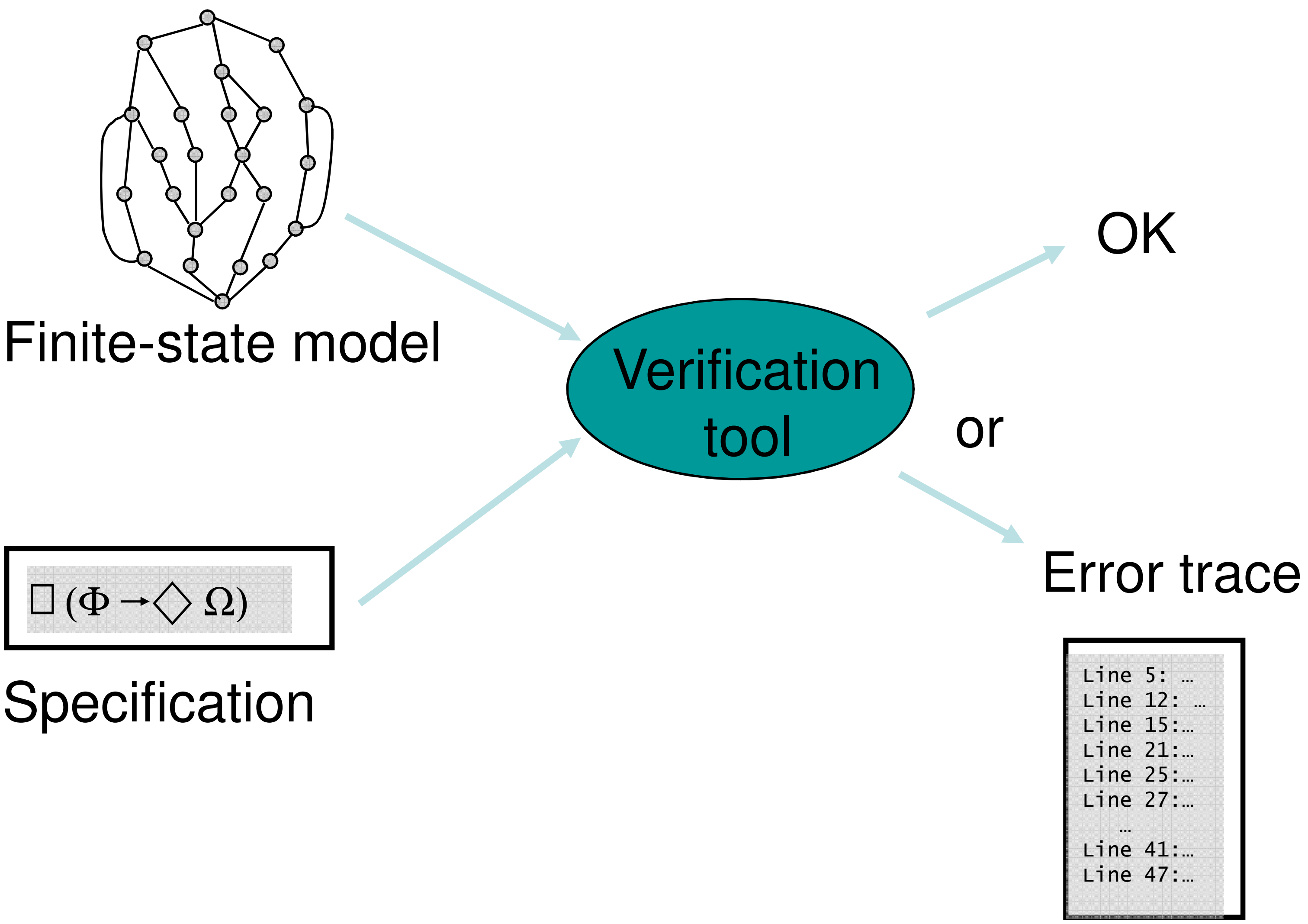
Temporal logic

- Qualitative vs. Quantitative
  - Something will happen in future
  - Something will happen after some specific time

Model checking

- **Process of Model Checking:**
  - Modeling
  - Specification
  - Verification Method

Basic picture of Model Checking



```
Line 5: …
Line 12: …
Line 15:…
Line 21:…
Line 25:…
Line 27:…
    …
Line 41:…
Line 47:…
```

where do we get the system model?

hardware

e.g., Verilog or VHDL, source code

abstraction & other (semi-)automated transformations

software

e.g., C, C++ , or Java, source code

state machine-based system model

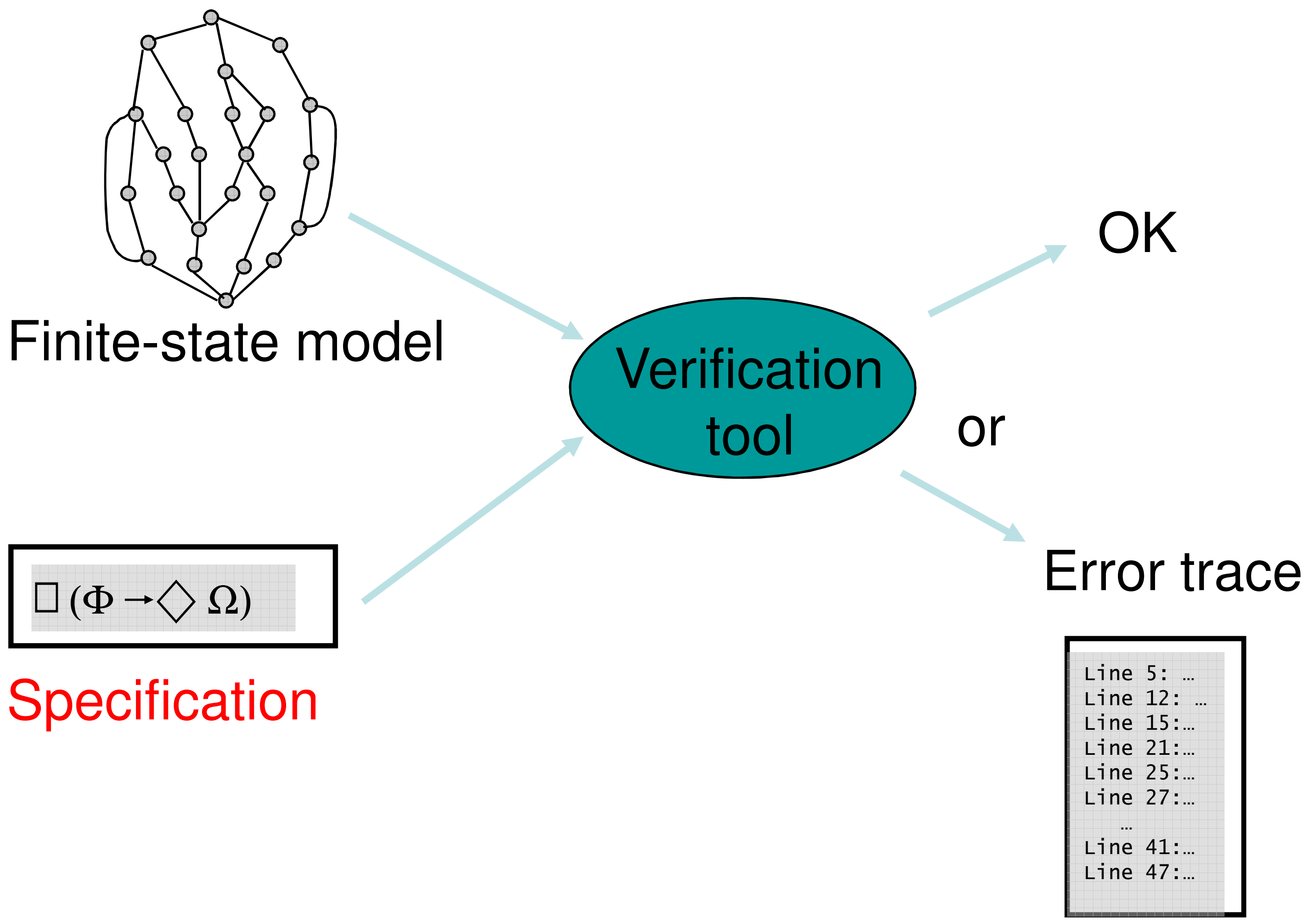
hand-built design models

Statecharts (Harel 1987)

CSP (Hoare 1985)
(Communicating Sequential Processes)

CCS (Milner 1980)
(Calculus of Communicating Systems)

Basic picture of Model Checking



```
Line 5:  …
Line 12:  …
Line 15:…
Line 21:…
Line 25:…
Line 27:…
   …
Line 41:…
Line 47:…
```

Questions

1. What are the problems with simulation based validation method.
2. Why Formal methods did not get acceptance in industry earlier.
3. What are the advantages of using formal methods for design verification.
4. Why it is difficult to use HOL in verification.
5. Try to find out major system design failure like Pentium Bug.

NPTEL Phase-II
Video course on

Design Verification and Test of Digital VLSI Designs

Dr. Santosh Biswas
Dr. Jatindra Kumar Deka
IIT Guwahati

Module IV: Temporal Logic

Lecture II: Temporal Logic:
Introduction and Basic Operators

Temporal Logic

- To capture timing behaviour
- Linear and Branching
- Discrete and Continuous
- Qualitative and Quantitative

Temporal Logic

- The truth value of a temporal logic is defined with respect to a model.
- Temporal logic formula is not *statically* true or false in a model.

Temporal Logic

- The models of temporal logic contain several states and a formula can be true in some states and false in others.

Temporal Logic

In temporal logic we can express statements like:

- "I am *always* happy",
- "I will *eventually* be happy",
- "I will be happy *until* I do something wrong“
- "I am happy."

Temporal Logic Operator

Temporal logic has two kind of operators:

- Logical operator
- Temporal operator

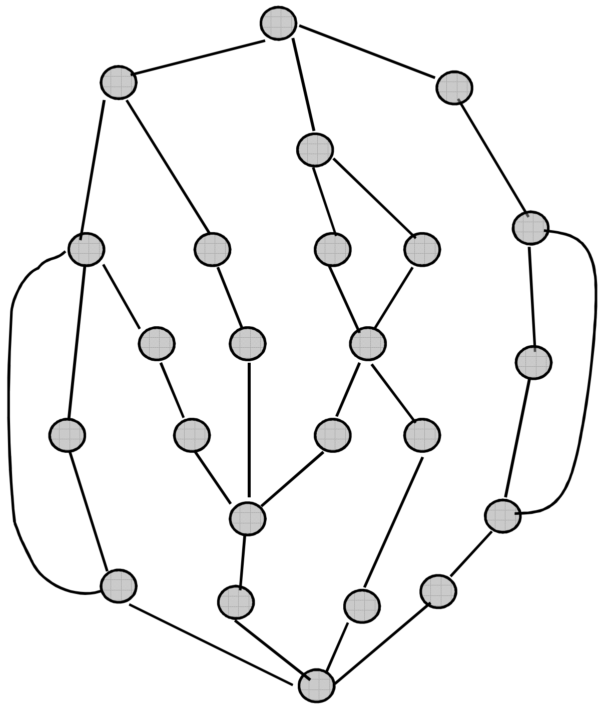
Temporal Operator

Operator	Textual Notation	Meaning
○	X φ	Φ holds at next state
◊	F φ	Φ eventually holds
□	G φ	Φ holds globally
U	φ **U** ψ	Φ holds until ψ holds

Temporal formulas are interpreted over a model, which is an infinite sequence of states.

Given a model *M* and a temporal formula φ , we define an inductive definition for the notion of φ holding at a position S_j in *M* and denoted by

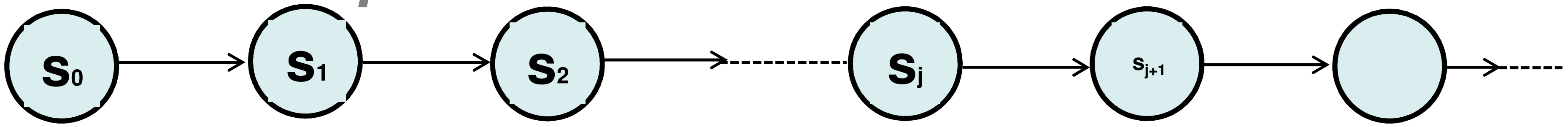
$(M, S_j) \models \varphi$



Next : X φ

$(M, S_j) \vDash X\, \boldsymbol{\varphi} \Leftrightarrow (M, S_{j+1}) \vDash \boldsymbol{\varphi}$

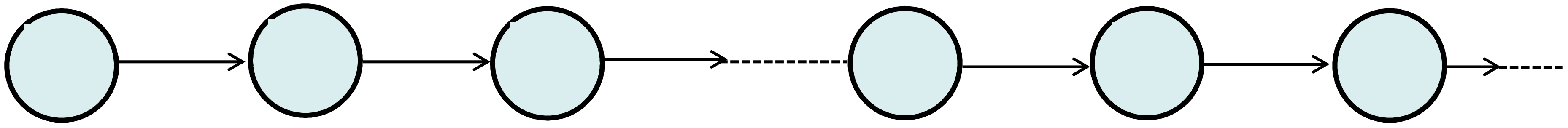
State S_j satisfies as its next state S_{j+1} satisfies φ .



Future : F φ

$(M, S_j) \vDash F\,\varphi \iff \exists k, k \geq j, (M, S_k) \vDash \varphi$

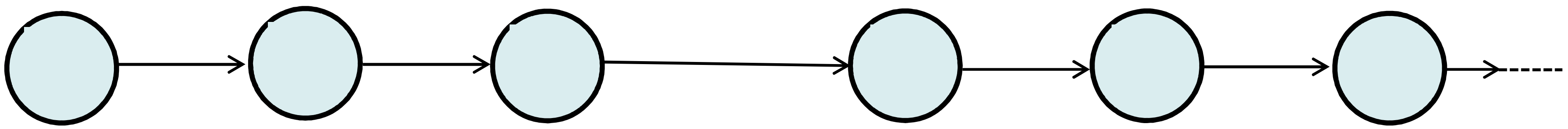
state S_j satisfies as future state S_k satisfies φ
.



Globally : G φ

$(M, S_j) \vDash G\varphi \Leftrightarrow \forall k, k \geq j, (M, S_k) \vDash \varphi$

S_j satisfies **G** φ as all states satisfies φ.

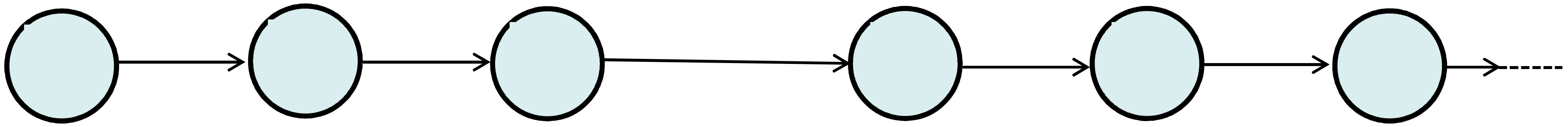


Until : $\mathbf{\Psi}$ **U** $\boldsymbol{\varphi}$

$$(M, S_j) \vDash \mathbf{\Psi} \cup \boldsymbol{\varphi} \Leftrightarrow \exists k, (M, S_K) \vDash \boldsymbol{\varphi} \text{ and}$$

$$\forall k, j < k\ (M, S_j) \vDash \mathbf{\Psi}$$

S_j satisfies **(**$\mathbf{\Psi}$ **U** $\boldsymbol{\varphi}$) because $\mathbf{\Psi}$ is true for all states S_i $j<=i<k$ and then $\boldsymbol{\varphi}$ is true for state S_k .



Temporal Operator

Future Logic

Operator	Textual Notation	Meaning
○	X φ	Φ holds at next state
◊	F φ	Φ eventually holds
□	G φ	Φ holds globally
U	φ **U** ψ	Φ holds until ψ holds

- Past Temporal Logic
  - Previous
  - Eventually in Past
  - Globally in Past
  - Back to

Previous: φ has to hold at the previous state.

$$(M, S_j) \models \text{\textcircled{$\sim$}}\ \boldsymbol{\varphi} \Leftrightarrow \exists\, (M, S_{j-1}) \models \boldsymbol{\varphi}$$

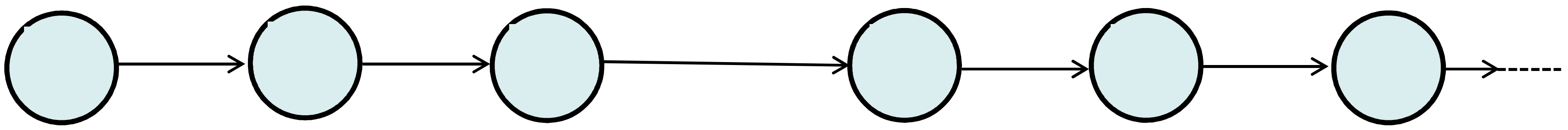
state S_j satisfies $\text{\textcircled{$\sim$}}$ φ as its previous state S_{j-1} satisfies φ .

$$\begin{tikzcd}
s_0 \arrow[r] & s_1 \arrow[r] & s_2 \arrow[r] & {} \arrow[r, dash, dashed] & s_{j-1} \arrow[r] & s_j \arrow[r] & \bigcirc \arrow[r] & {} \arrow[r, dash, dashed] & {}
\end{tikzcd}$$

Eventually in past: φ eventually has to hold in the past.

$(M, S_j) \models \Diamond\!\!\!\!\!\sim\ \boldsymbol{\varphi} \Leftrightarrow \exists k, k \leq j (M, S_k) \models \boldsymbol{\varphi}$

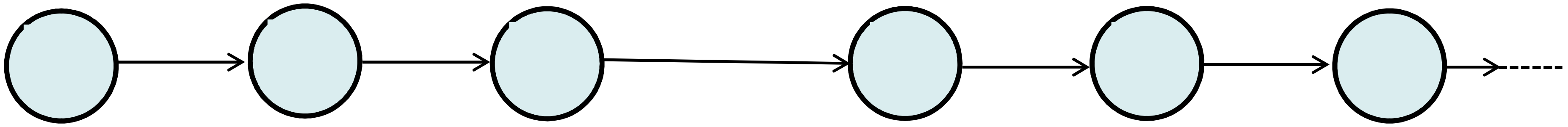
state S_j satisfies $\Diamond\!\!\!\!\!\sim\ \boldsymbol{\varphi}$ as eventually a past state S_k satisfies $\boldsymbol{\varphi}$.



Globally in past: φ has to hold on the entire previous path.

$(M, S_j) \vDash \boxed{\sim} \quad \varphi \Leftrightarrow \forall k, k \leq j (M, S_k) \vDash \varphi$

state S_j satisfie $\boxed{\sim}$ φ, as globally in all past states staring backward from S_j , satisfies φ.

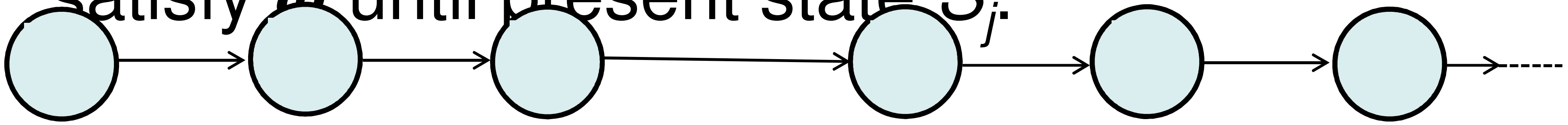


Back to: φ holds in all previous states (including the present) starting at the last position Ψ held.

(M, S_j) ⊨ $\varphi\beta\Psi \Leftrightarrow \exists k$(M, S_k) ⊨Ψ and $\forall j \geq k$ (M, S_j) ⊨ $\boldsymbol{\varphi}$ until present state

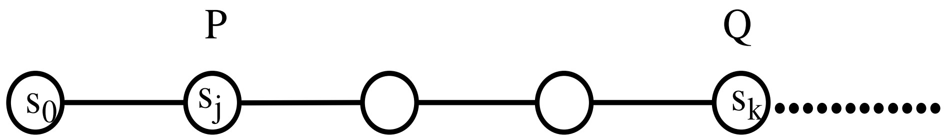
OR (M, S_j) ⊨ $\boldsymbol{\varphi}$ for j=0 to present state

in state S_k Ψ is true and for all the states satisfy $\boldsymbol{\varphi}$ until present state S_j.



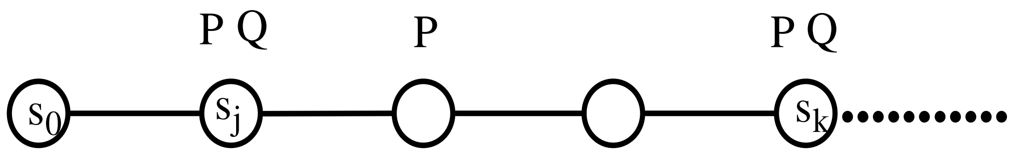
Example

($P \rightarrow$ FQ): If P is true in a state then in a future state Q is true.



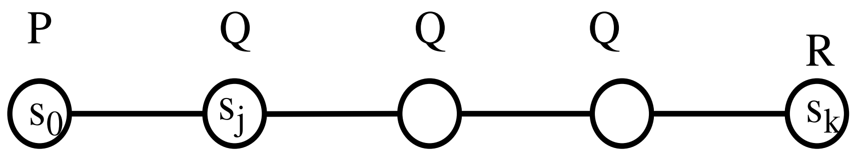
Examples

(P* ∨ *XQ): Either *P* holds in a state or in next state *Q* holds



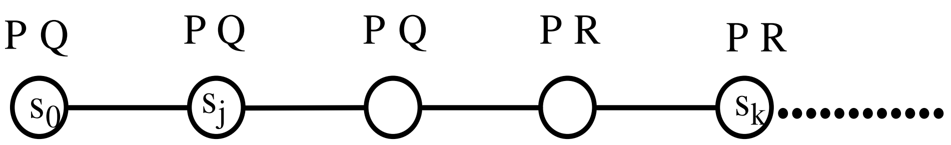
Examples

(P* ∨ *(Q* U *R): Either *P* holds in a state or *Q* U *R* (*Q* until *R*) holds



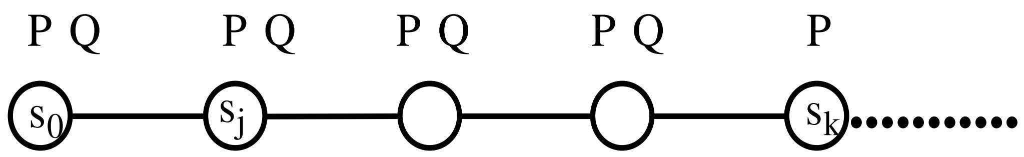
Examples

***(P* $\wedge$ *(Q* U *R)*:** *P* holds in a state and also *Q* U *R* (*Q* until *R*) holds in the state



Examples

$(P \wedge \overset{\sim}{\bigcirc} Q)$: *P* holds in a state and in the previous state *Q* holds



Questions

What does the temporal formula $(P \mathbin{\overset{\sim}{\Diamond}} Q)$ mean? Give an example where this formula is valid in all the states.

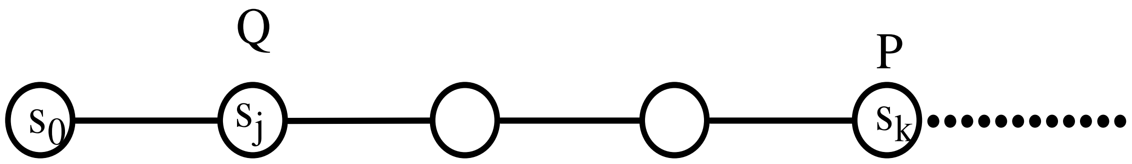
The temporal operator used here is Eventually in Past

Questions

What does the temporal formula $(P \overset{\sim}{\Diamond} Q)$ mean? Give an example where this formula is valid in all the states.

The temporal operator used here is Eventually in Past

$(P \rightarrow \overset{\sim}{\Diamond} Q)$ means that "If *P* holds in a state then eventually in past *Q* holds".


Q
P
s_0
s_j
s_k

Questions

- Express the following information in temporal logic
  - P is true in next state, or the next but one.

Questions

- Express the following information in temporal logic
  - P is true in next state, or the next but one.

$X\, p \vee XXp$

Questions

- Express the following information in temporal logic
  - p is true in next state, or the next but one.

$X\,p \vee XXp$

Consider now: p is true in next state and the next but one.

Questions

- Consider the fact: *p* is an atomic proposition. Write the temporal formula for *p* is infinitely often true.

Questions

- Consider the fact: p is a atomic proposition. Write the temporal formula for p is infinitely often true.
  - G F p

Questions

- Consider the fact: p is a atomic proposition. Write the temporal formula for p is infinitely often true.

  – G F p

Give a model to show that this formula is true in all states.

NPTEL Phase-II
Video course on

Design Verification and Test of Digital VLSI Designs

Dr. Santosh Biswas
Dr. Jatindra Kumar Deka
IIT Guwahati

Module IV: Temporal Logic

Lecture III: Syntax and Semantics of CTL

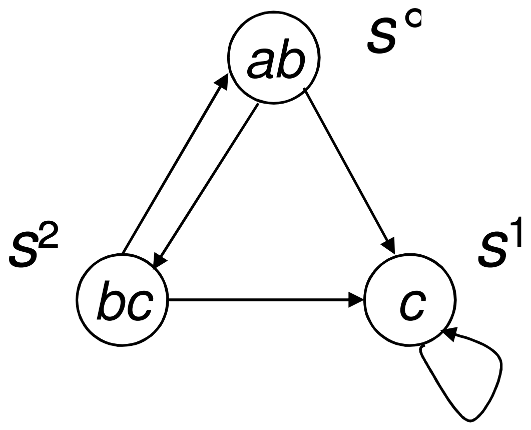
Temporal Logic

- Temporal Logic
  - Meaning is defined over a model.
- Given a model *M* and a temporal formula φ , we define an inductive definition for the notion of φ holding at a position S_j in *M* and denoted by $(M, S_j) \models \varphi$

- Type of Formulas
  - Path Formulas
  - State formulas

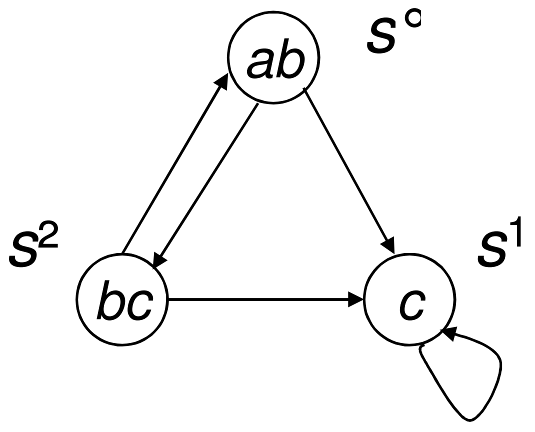
Temporal Logic

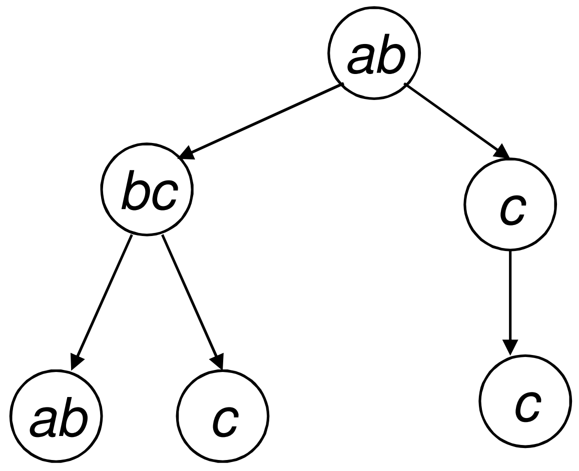
- Computational Tree Logic (Branching Time Logic)
  - Meaning is defined over a model.



Temporal Logic

- Computational Tree Logic (Branching Time Logic)
  - Meaning is defined over a model.





Syntax of CTL

A CTL formula comprises

1. Atomic propositions such as *{p, q, r…..}*
2. Path Quantifiers *{A,E}*
    a. *A* : all paths starting from a given state.
    b. *E* : there exists at least one path from a given state.
3. Propositional logic operators such as AND ($\wedge$), OR ($\vee$), NOT ($\neg$)
4. Temporal operators *{X,F,G,U}*
    a. NE**X**T: next states of current state.
    b. **F**UTURE: any one of future states from the current state.
    c. **G**LOBAL: all future states from the current state.
    d. **U**NTIL: Some CTL formula holds until another CTL formula, from the current state.

We can define CTL formulas as:

$\Phi ::= \bot \mid \top \mid P \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi\ U\ \varphi] \mid E[\varphi\ U\ \varphi];$

where

- The symbol $\top$ means truth value 'true' and symbol $\bot$ means truth value 'false'.
- *P* ranges over a set of atomic propositions

Let V be a set of atomic propositions

CTL formulas are defined recursively:

Every atomic proposition is a CTL formula

If f_1 and f_2 are CTL formulas, then so are $\neg f_1$,
$f_1 \wedge f_2$,
AX f_1, EX f_1, A[f_1 U f_2] and E[f_1 U f_2], AGf_1,
EGf_1, AFf_1, EFf_1

AX f_1 means: holds in state $s°$ iff f_1 holds in all successor states of $s°$

EX f_1 means: There exists a successor such that f_1 holds

A[f_1 U f_2] means: always until, in all paths such that f_1 holds until is f_2 satisfied.

E[f_1 U f_2] means: There exists a path such that f_1 holds until is f_2 satisfied.

AGf_1: Always globally f_1 holds.

EGf_1: There exists a path where f_1 holds globally

AFf_1: f_1 holds in all path in future.

EFf_1: There exists a path in which f_1 holds in future.

- In CTL, every temporal operator must be preceded by a path quantifier.
  - State formula

Examples

- $AG(p \rightarrow \neg EG\neg q)$
- $EGp \quad E(q\ U\ r)$
- $AG\neg(p \wedge \quad q)$
- $AG\neg(EF\ p \wedge \quad q)$
- $AF\ EG\ p$
- $A[p\ U\ A[q\ U\ r]]$
- $A[AX\neg p\ U\ EX(\neg p \quad q)] \rightarrow A[p\ U\neg q]$

Examples

- Gp
- $EFGr$
- $F[r \ U \ q]$
- $AEFr$
- $A[(r \ U \ q) \wedge (p \ U \ r)]$

Temporal structures

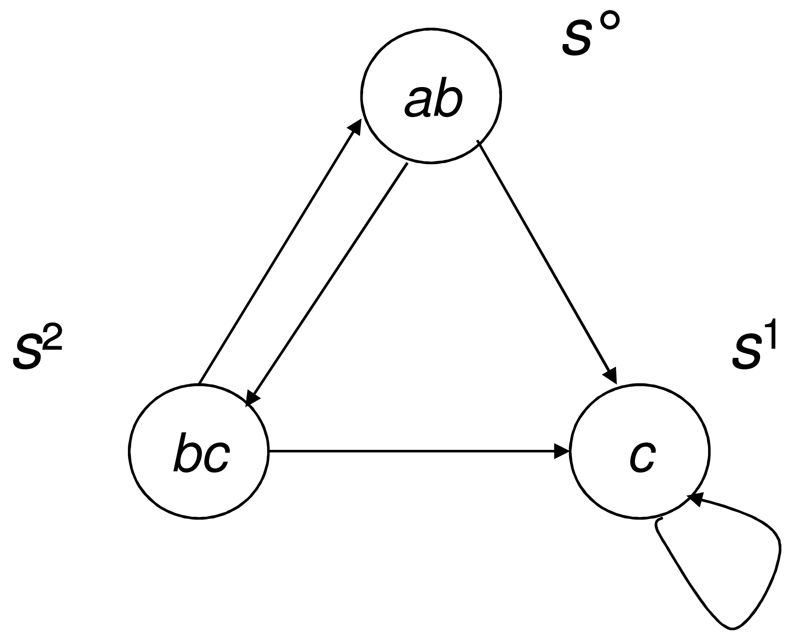
- The semantics of CTL is defined over a model M, which is defined as 3-tuple $M = (S, \rightarrow, L)$

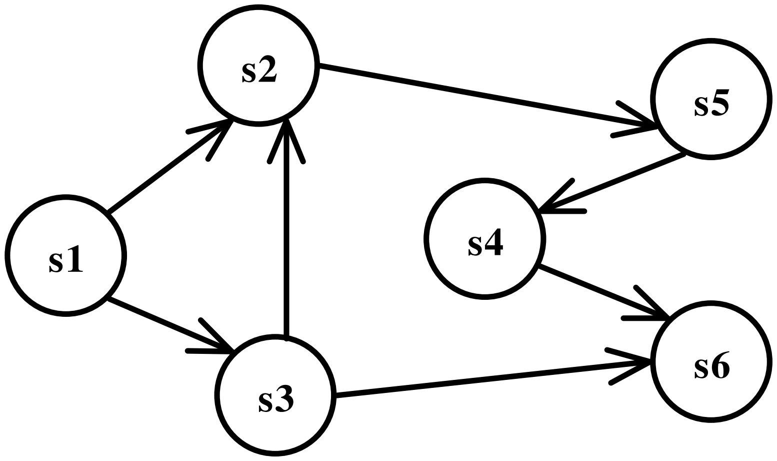
- **Definition**: A **temporal structure** $M := (S, \rightarrow, L)$ consists of
  1. A finite set of states S
  2. A transition relation $\rightarrow \subseteq S \times S$ with $\forall s \in S\ \exists s' \in S : (s, s') \in \rightarrow$
  3. A labeling function $L: S \rightarrow \wp(V)$, with being the set of propositional variables (atomic formulas)

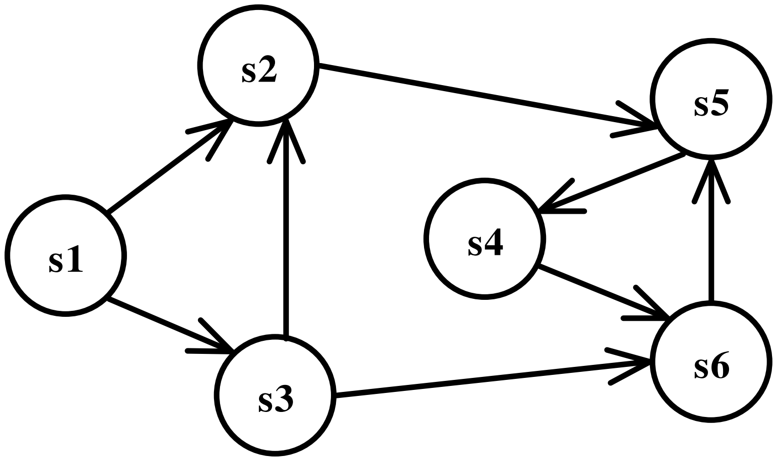
- This structure is often called Kripke structure.

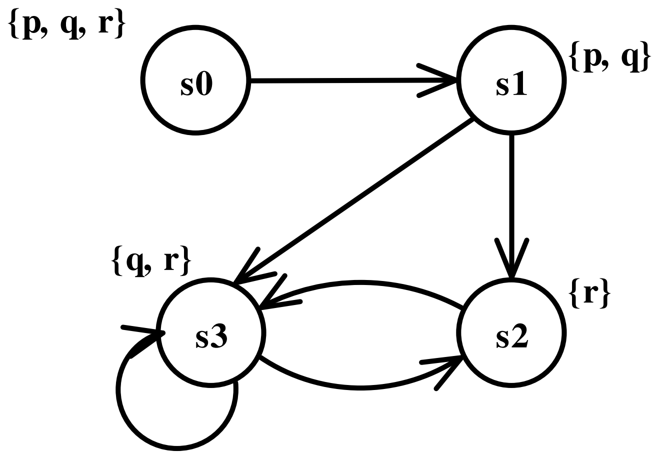
Semantics of CTL

- This model is also known as Kripke structure.
- A Kripke structure is similar to a state transition diagram, with
  - All states must have at least one outgoing edge.
  - Each state is labeled with one of the element of the power set of atomic propositions.


s°
ab
s^2
bc
s^1
c


s1
s2
s3
s4
s5
s6


s1
s2
s3
s4
s5
s6



- S={s0,s1,s2,s3}
- $\rightarrow$ ={{s0, s1}, {s1,s2}, {s1, s3}, {s2,s3}, {s3,s2},{s3,s3}}.
- L: L(s0)={p,q,r}, L(s1)={p, q}, L(s2)={r}, L(s3)={q, r}.

CTL Semantics

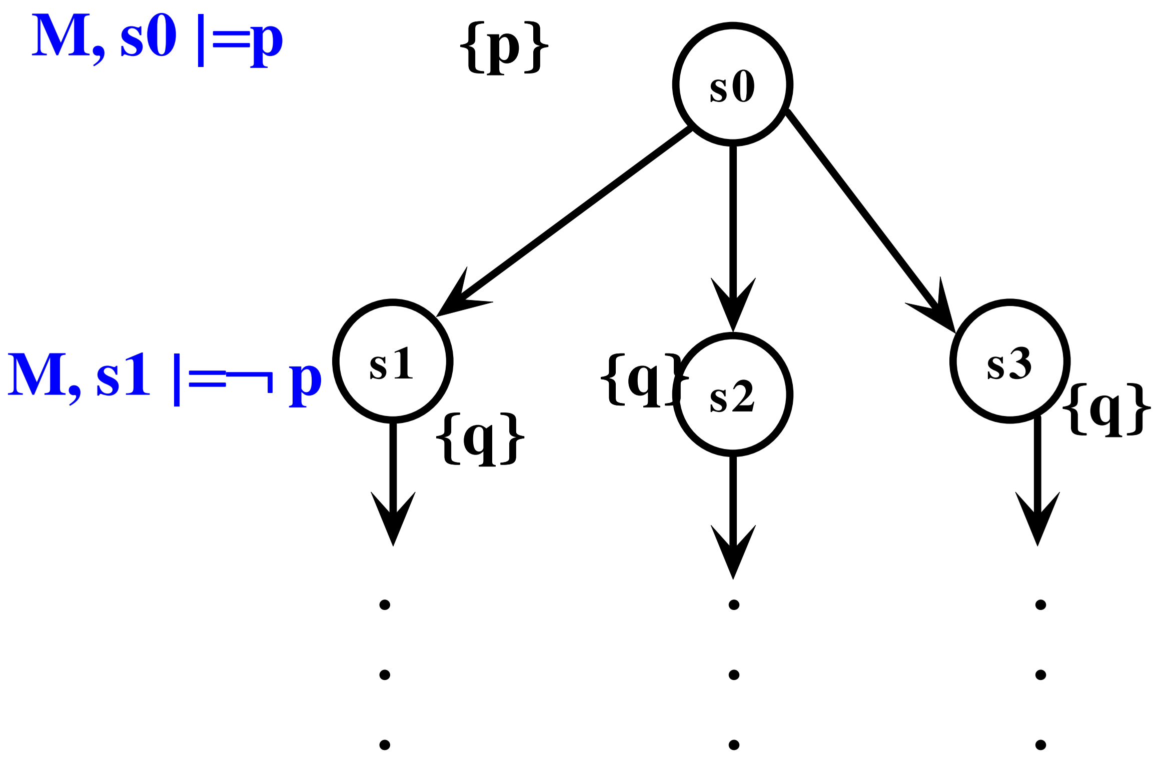
Let $M = (S, \rightarrow, L)$ be a model for CTL. Given any s in S, we define whether a CTL formula Φ holds in state s. We denote this by $M, s \models \Phi$

The relation M, $s \models \varphi$ is defined by structural induction on φ, as follows

$M, s \models \top$ and $M, s \not\models \bot$;

$M, s \models p$ iff $p \in L(s)$; atomic proposition p is satisfied if label of s has p.

$M, s \models \neg\varphi$ iff $M, s \not\models \varphi$. $\neg\varphi$ is satisfied at s if s does not satisfy φ.



L(s0)={p}

L(s1)={q}

L(s2)={q}

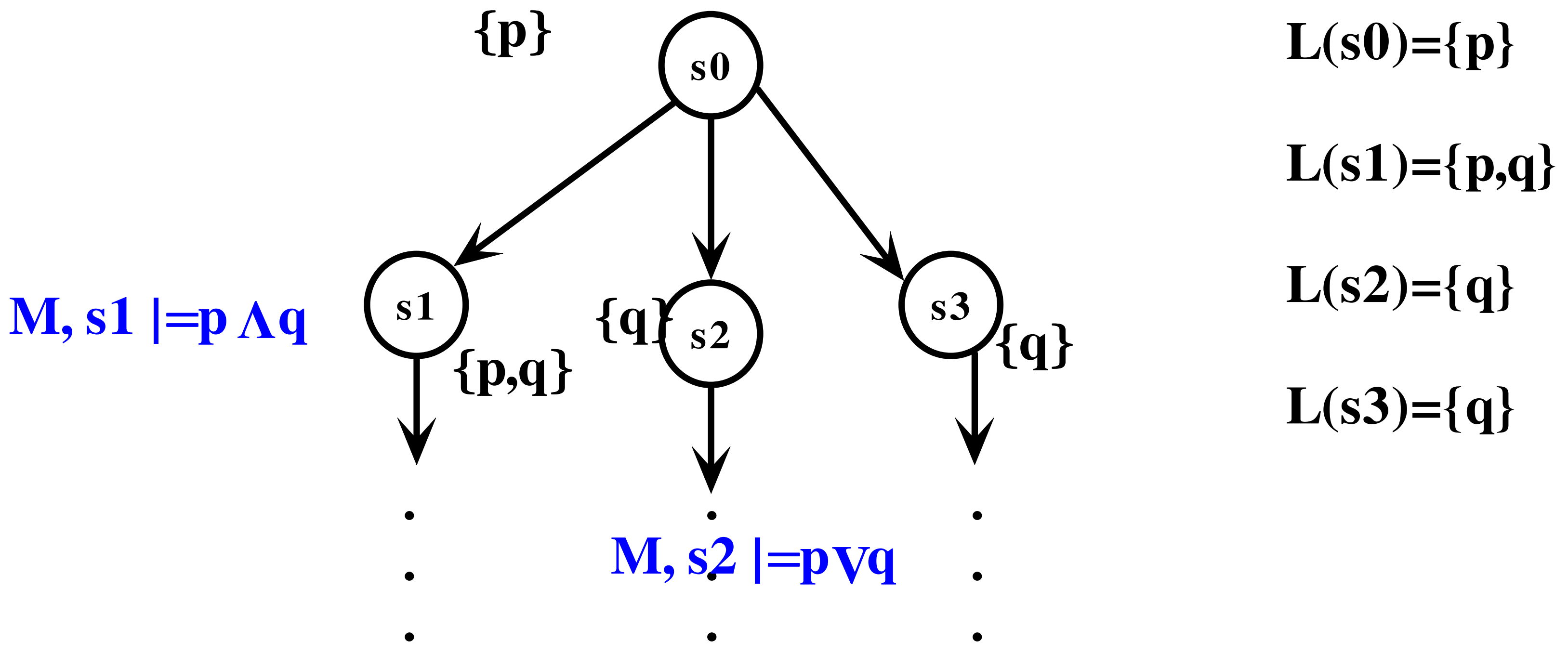
L(s3)={q}

$M, s \models \varphi 1 \wedge \varphi 2$ iff $M, s \models \varphi 1$ and $M, s \models \varphi 2;$

$\varphi 1 \wedge \varphi 2$ *i*s satisfied at *s* if in *s* both $\varphi 1$ and $\varphi 2$ are satisfied.

$M, s \models \varphi 1 \vee \varphi 2$ iff $M, s \models \varphi 1$ or $M, s \models \varphi 2;$

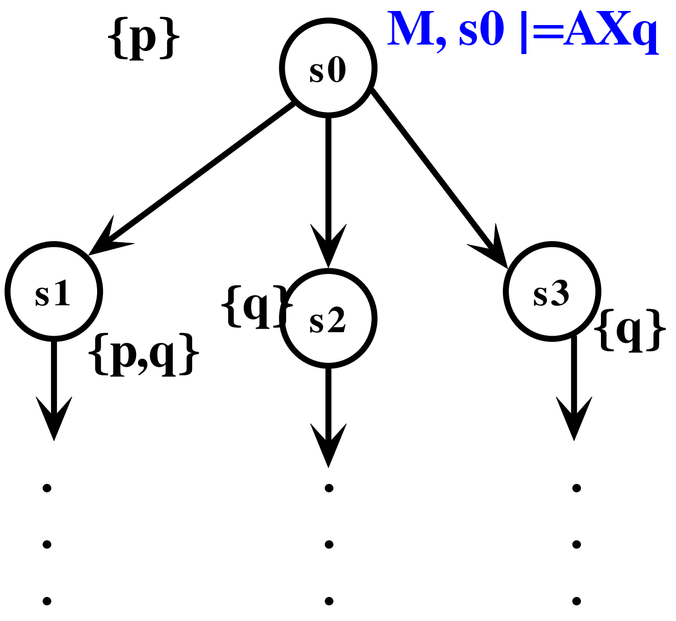
$\varphi 1 \vee \varphi 2$ is satisfied at *s* if in *s* either $\varphi 1$ or $\varphi 2$ is satisfied.



$M, s \models \varphi 1 \rightarrow \varphi 2$ iff $M, s \not\models \varphi 1$ or $M, s \models \varphi 2$;

$\varphi 1 \rightarrow \varphi 2$ is satisfied at s if in s either $\varphi 1$ is not satisfied or $\varphi 2$ is satisfied.

$M, s \models AX\varphi$ iff for all s1 such that s $\longrightarrow$ s1, we have M, s1 $\models \varphi$; $AX\varphi$ is satisfied at s if in all next states of s, φ is satisfied.



L(s0)={p}

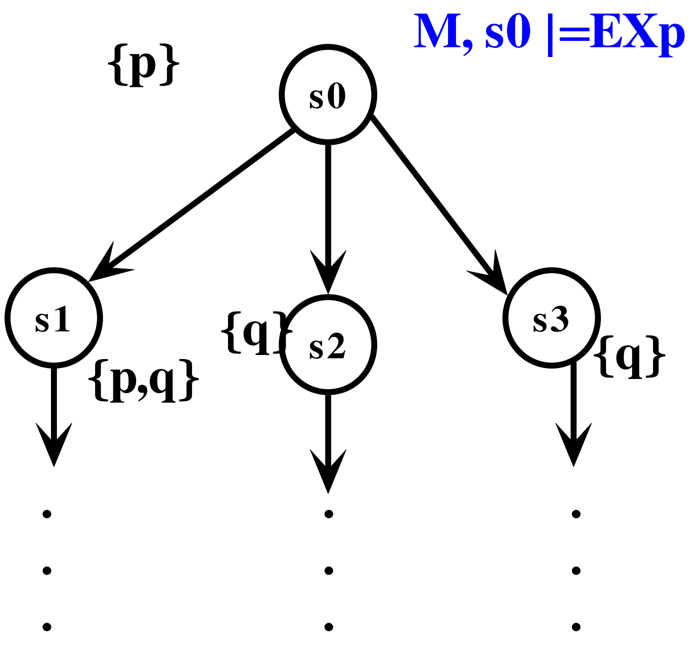
L(s1)={p,q}

L(s2)={q}

L(s3)={q}

$M, s \models EX\varphi$ iff for one state s1 such that s → s1 we have M, s1 $\models \varphi$; $EX\varphi$ is satisfied at s, if in some next state of s, φ is satisfied.

s0 $\models EXp$.



L(s0)={p}

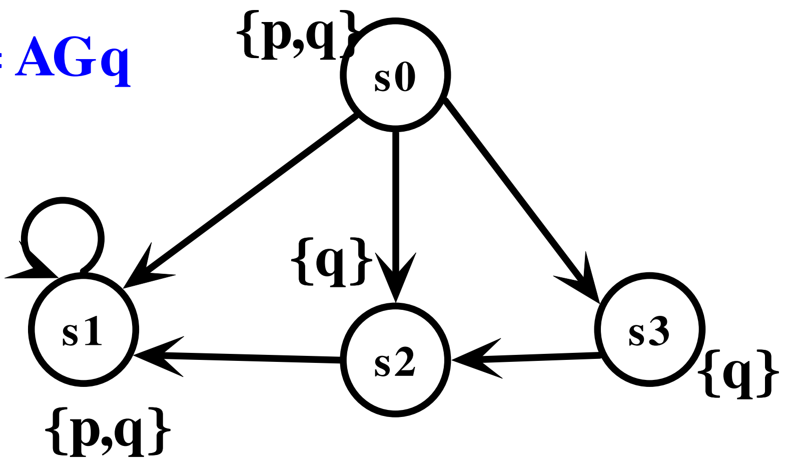
L(s1)={p,q}

L(s2)={q}

L(s3)={q}

$M, s \models AG\varphi$ holds iff for all paths s1 → s2 → s3 → . . ., where s=s1, and all s_i along the path, $M, s_i \models \varphi$. $AG\varphi$ is satisfied at s if all states of all paths from s satisfies φ.

M, s0 |= AGq



L(s0)={p,q}

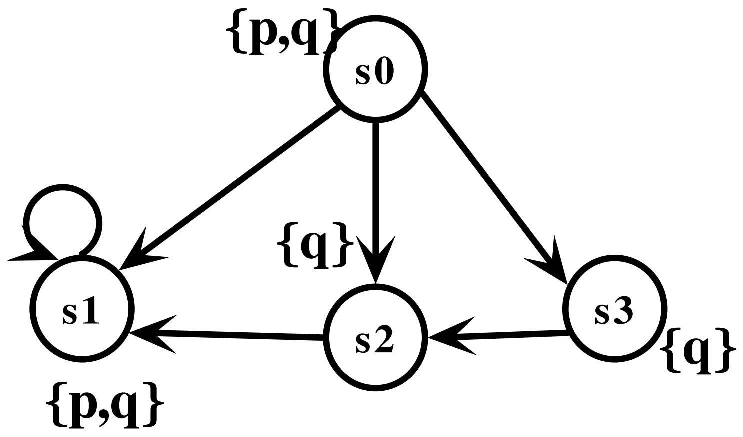
L(s1)={p,q}

L(s2)={q}

L(s3)={q}

$M, s \models EG\varphi$ holds iff there is a path s1 → s2 → s3 → . . ., where s=s1, and all s_i along the path, $M, s_i \models \varphi$. $EG\varphi$ is satisfied at s if all states of at least one path from s satisfies φ.

state s0 satisfies EGp (the path s0 → s1 → s1 → ….).



L(s0)={p,q}

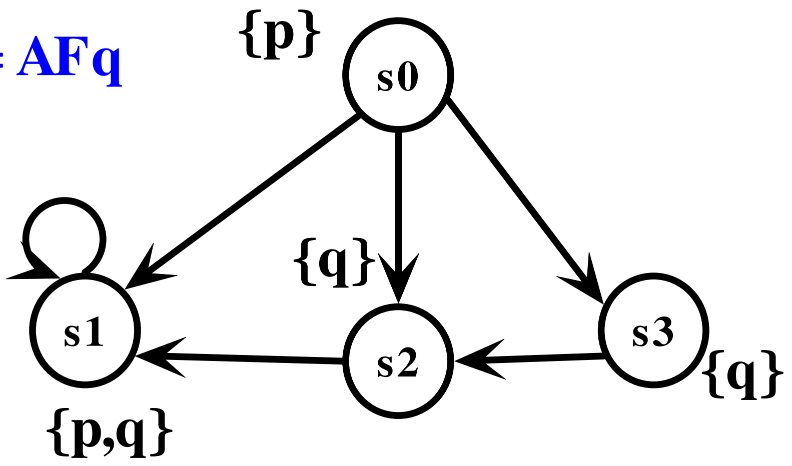
L(s1)={p,q}

L(s2)={q}

L(s3)={q}

$M, s \models AF\varphi$ holds iff for all paths s1 → s2 → s3 → . . ., where s=s1, and for at least one s_i along the path, $M, s_i \models \varphi$. $AF\varphi$ is satisfied at s if some "future" state of all paths from s satisfies φ.

M, s0 |= AFq



L(s0)={p}

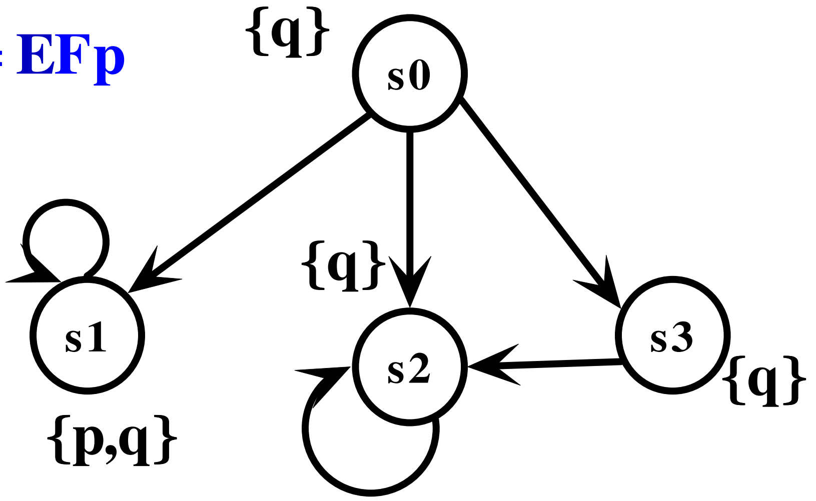
L(s1)={p,q}

L(s2)={q}

L(s3)={q}

$M,\ s \models EF\varphi$ holds iff there is one path s1 $\rightarrow$ s2 $\rightarrow$ s3 $\rightarrow$ . . ., where s=s1, and for at least one s_i along the path, $M,\ s_i \models \varphi$.

M, s0 |= EFp



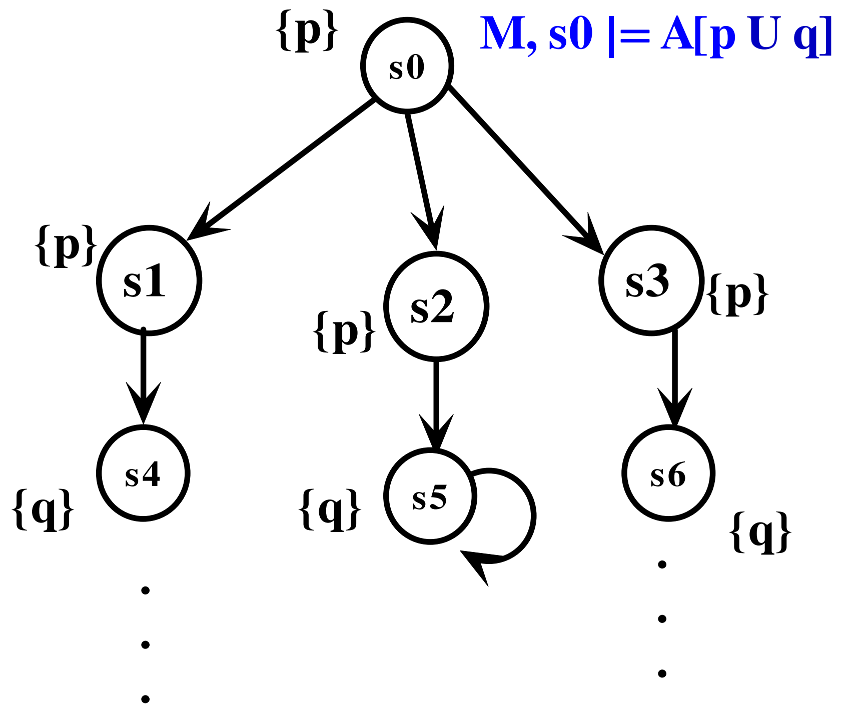
L(s0)={q}

L(s1)={p,q}

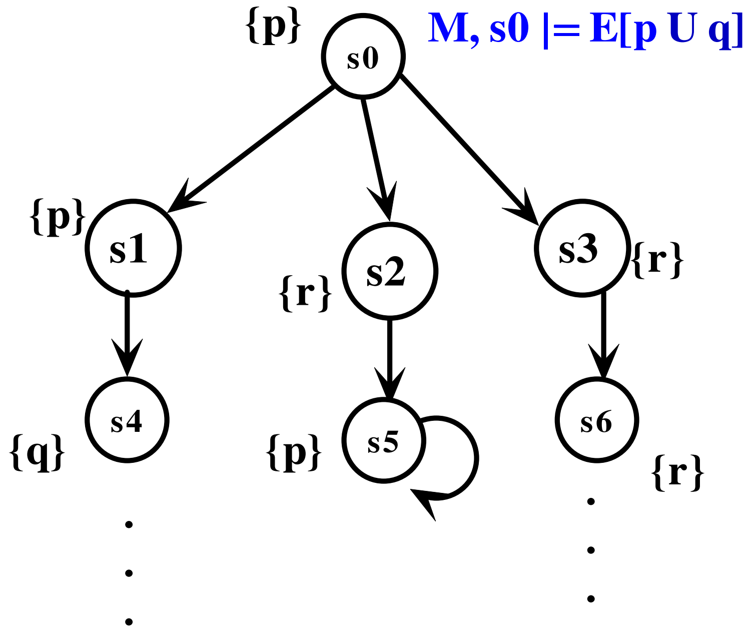
L(s2)={q}

L(s3)={q}

M, s $\models$ *A[φ1 U φ2]* holds iff for all paths s1 → s2 → s3 → . . ., where *s*=s1, *φ1 U φ2* is satisfied, i.e., there is some s_i along the path, such that *M,* $s_i \vdash$ *φ2*, and for each $j < i$, we have *M,* $s_j \models$ *φ1*



M, *s* ⊨ *E[φ1 U φ2]* holds iff for at least one path s1 → s2 → s3 → . . ., where *s*=s1, *φ1 U φ2* is satisfied, i.e., there is some s_i along the path, such that *M*, s_i ⊨ *φ2*, and for each $j < i$, we have *M*, s_j ⊨ *φ1*



Questions

- Consider X = {p, q, r} be a set of atomic proposition. What is the power set of X.

Questions

Show a Kripke structure such that in a particular state EX (q or r) holds but EX(q and r) does not hold.

Questions

Show a Kripke structure such that in a particular state AF (q or r) holds but EF(q and r) does not hold.

Questions

Express the following property in CTL:

It is possible to get a state where started holds, but ready does not hold.

Questions

Express the following property in CTL:

It is possible to get a state where started holds, but ready does not hold.

$$EF(\text{started} \wedge \neg \text{ready})$$

Questions

Express the following property in CTL:

For any state, if a request (of some resource) occurs, then it will eventually be acknowledged.

Questions

Express the following property in CTL:

For any state, if a request (of some resource) occurs, then it will eventually be acknowledged.

$$AG(\text{requested} \rightarrow AF\ \text{acknowledged})$$

NPTEL Phase-II
Video course on

Design Verification and Test of Digital VLSI Designs

Dr. Santosh Biswas
Dr. Jatindra Kumar Deka
IIT Guwahati

Module IV: Temporal Logic

Lecture IV: Syntax and Semantics of CTL – Continued

We can define CTL formulas as:

$\Phi ::= \bot \mid \top \mid P \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid AX\varphi \mid EX\varphi \mid AF\varphi \mid EF\varphi \mid AG\varphi \mid EG\varphi \mid A[\varphi\ U\ \varphi] \mid E[\varphi\ U\ \varphi];$

where

- The symbol $\top$ means truth value ‘true’ and symbol $\bot$ means truth value ‘false’.
- *P* ranges over a set of atomic propositions

Temporal structures

- The semantics of CTL is defined over a model M, which is defined as 3-tuple $M = (S, \rightarrow, L)$

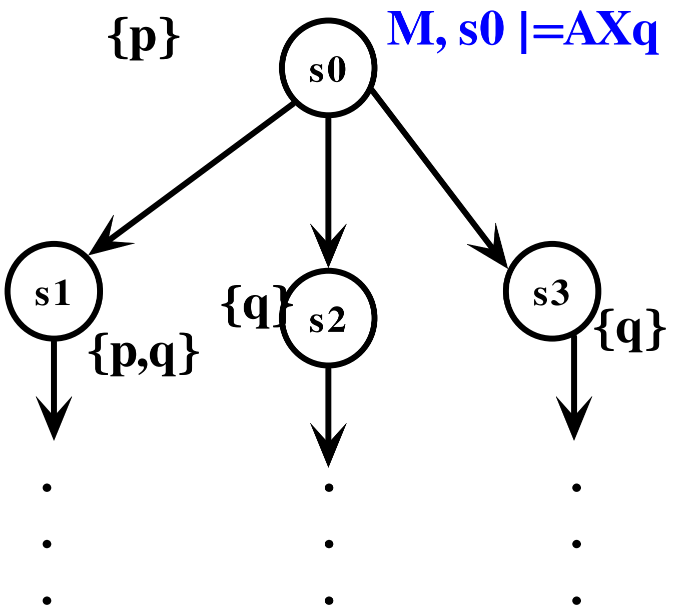
- **Definition**: A **temporal structure** $M := (S, \rightarrow, L)$ consists of

1. A finite set of states S
2. A transition relation $\rightarrow \subseteq S \times S$ with $\forall s \in S\ \exists s' \in S: (s,s') \in \rightarrow$
3. A labeling function $L: S \rightarrow \wp(V)$, with being the set of propositional variables (atomic formulas)

- This structure is often called Kripke structure.

$M, s \models AX\varphi$ iff for all s1 such that s $\longrightarrow$ s1, we have M, s1 $\models \varphi$;

$AX\varphi$ is satisfied at s if in all next states of s, φ is satisfied.



L(s0)={p}

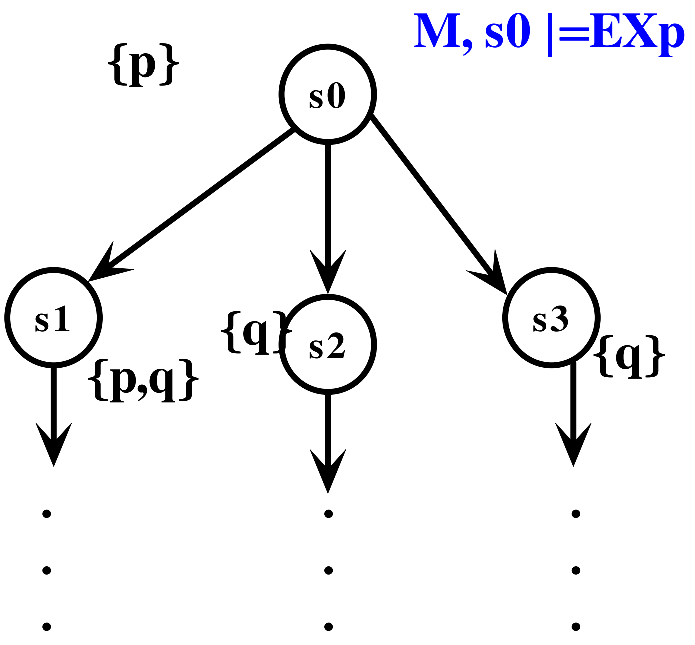
L(s1)={p,q}

L(s2)={q}

L(s3)={q}

$M, s \models EX\varphi$ iff for one state s1 such that s $\rightarrow$ s1 we have M, s1 $\vdash \varphi$;

$EX\varphi$ is satisfied at s, if in some next state of s, φ is satisfied.



L(s0)={p}

L(s1)={p,q}

L(s2)={q}

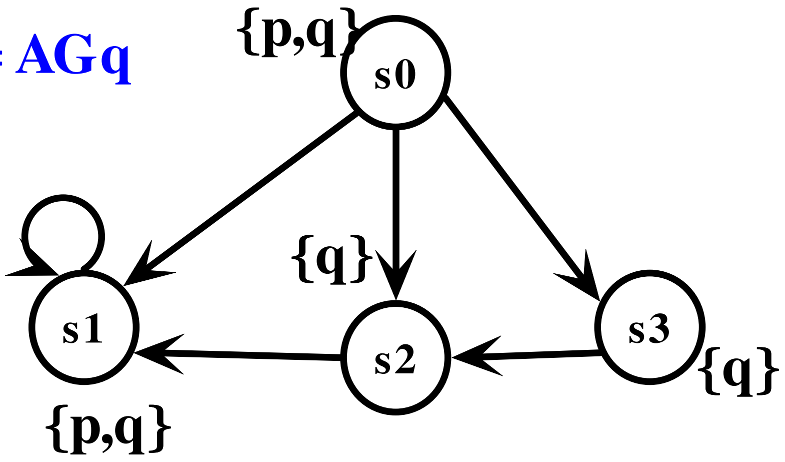
L(s3)={q}

$M, s \models AG\varphi$ holds iff for all paths s1 → s2 → s3 → . . ., where s=s1, and all s_i along the path,

$M, s_i \models \varphi$.

$AG\varphi$ is satisfied at s if all states of all paths from s satisfy φ.

M, s0 |= AGq



L(s0)={p,q}

L(s1)={p,q}

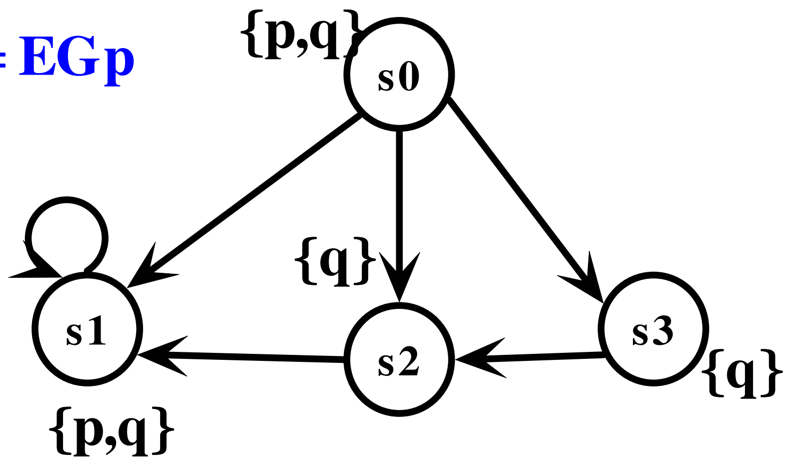
L(s2)={q}

L(s3)={q}

$M, s \models EG\varphi$ holds iff there is a path s1 → s2 → s3 → . . ., where s=s1, and all s_i along the path, $M, s_i \models \varphi$.

$EG\varphi$ is satisfied at s if all states of at least one path from s satisfies φ.

M, s0 |= EGp



L(s0)={p,q}

L(s1)={p,q}

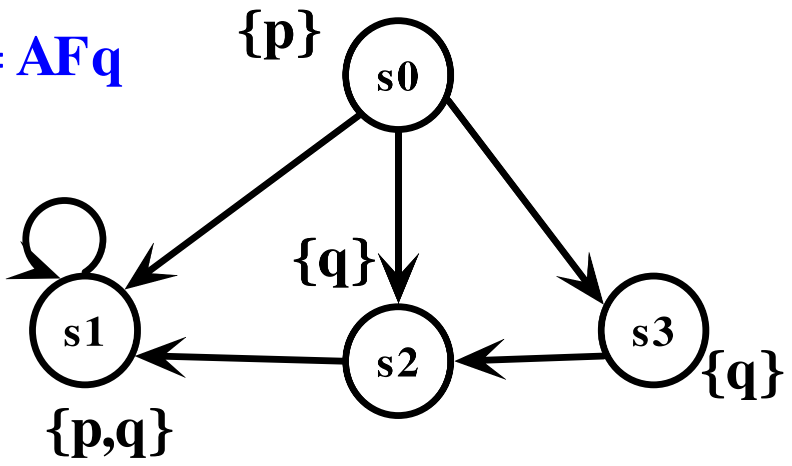
L(s2)={q}

L(s3)={q}

$M, s \models AF\varphi$ holds iff for all paths s1 → s2 → s3 → . . ., where s=s1, and for at least one s_i along the path, $M, s_i \models \varphi$.

$AF\varphi$ is satisfied at s if some "future" state of all paths from s satisfies φ.

M, s0 |= AFq



L(s0)={p}

L(s1)={p,q}

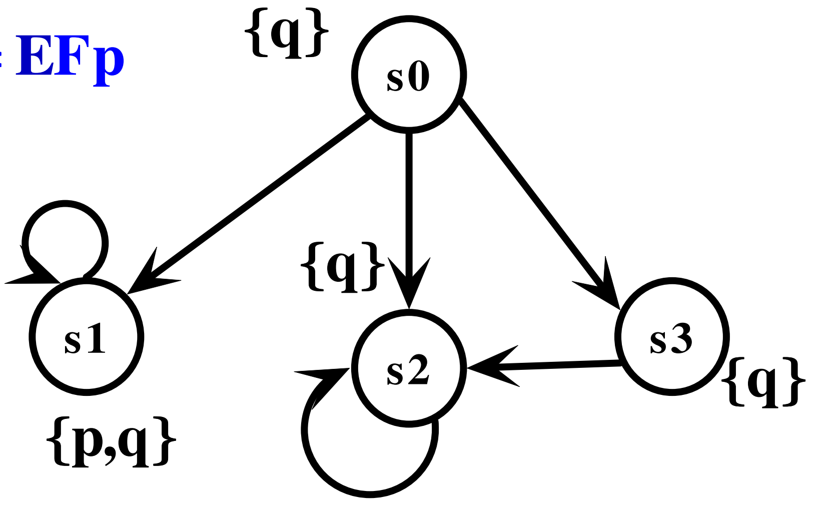
L(s2)={q}

L(s3)={q}

$M, s \models EF\varphi$ holds iff there is one path s1 $\longrightarrow$ s2 $\longrightarrow$ s3 $\longrightarrow$ . . ., where s=s1, and for at least one s_i along the path, $M, s_i \models \varphi$.

$EF\varphi$ is satisfied at s if some "future" state of all paths from s satisfies φ.

M, s0 |= EFp



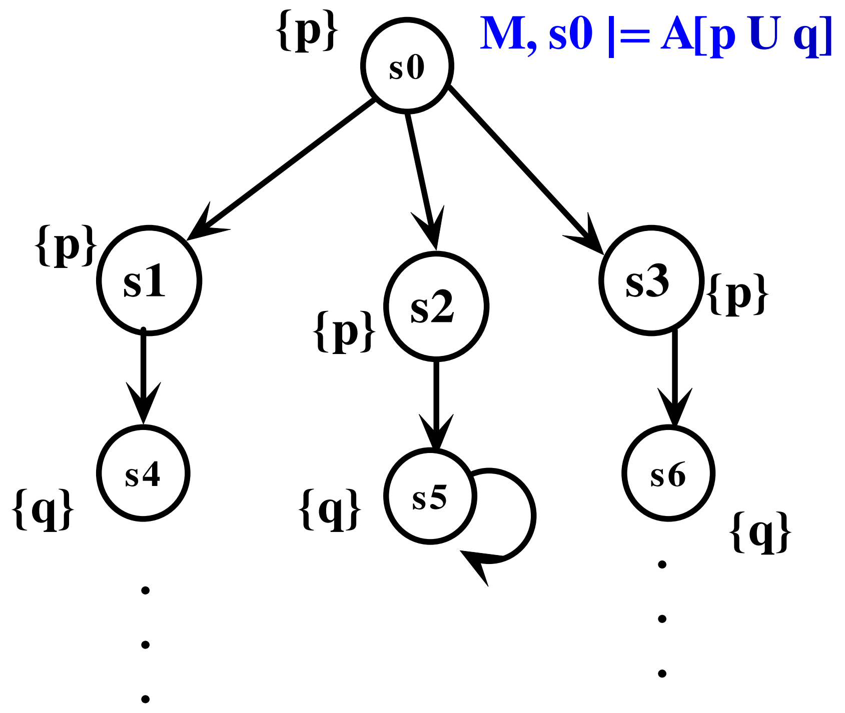
L(s0)={q}

L(s1)={p,q}

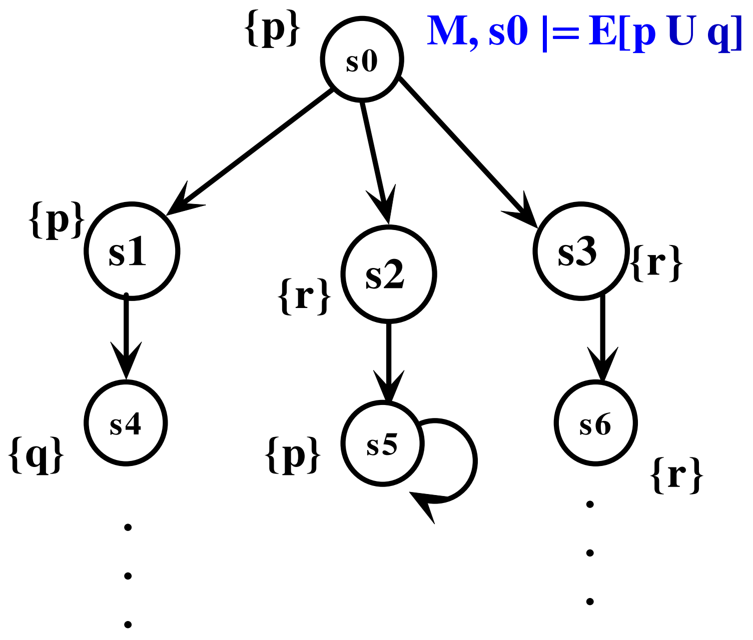
L(s2)={q}

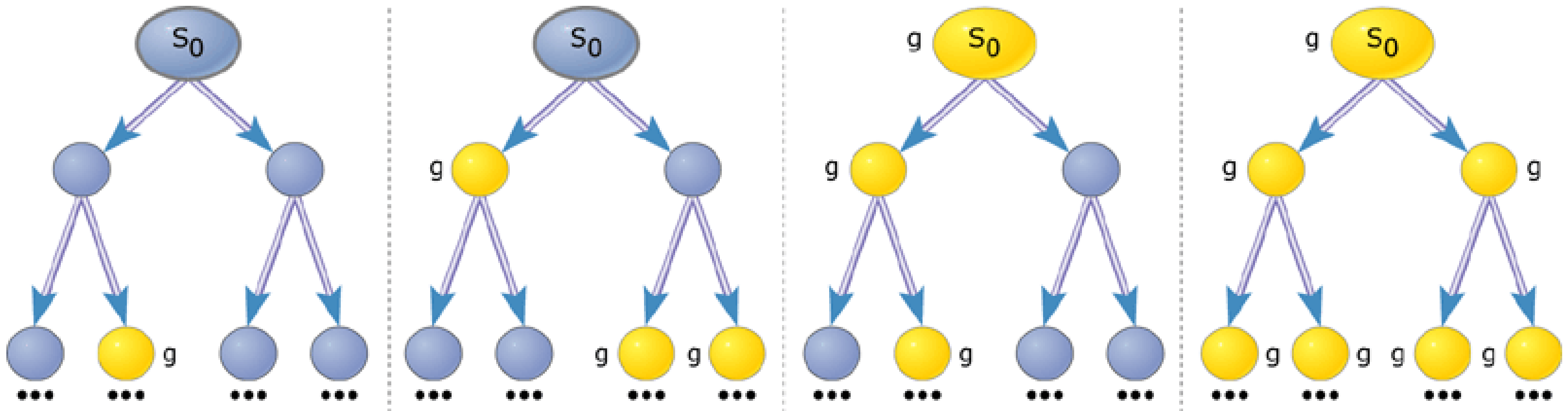
L(s3)={q}

$M, s \models A[\varphi 1\ U\ \varphi 2]$ holds iff for all paths s1 → s2 → s3 → . . ., where s=s1, $\varphi 1\ U\ \varphi 2$ is satisfied, if there is some s_i along the path, such that $M,\ s_i \models \varphi 2$, and for each $j < i$, we have $M,\ s_j \models \varphi 1$

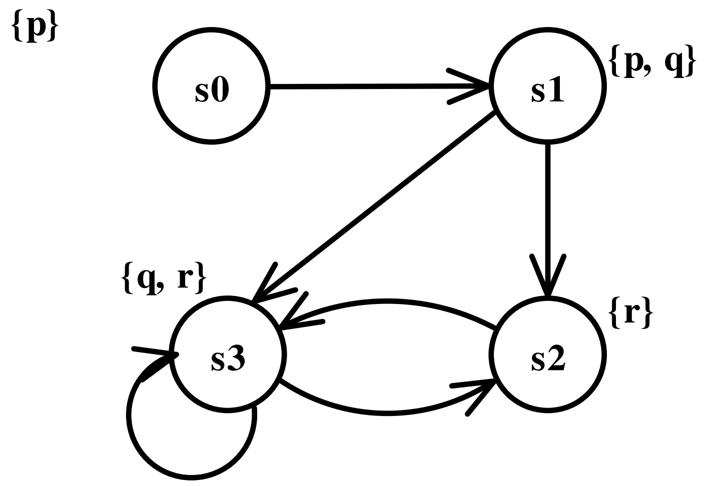


M, s $\models$ *E[φ1 U φ2]* holds iff for at least one path s1 → s2 → s3 → . . ., where *s*=s1, *φ1 U φ2* is satisfied, if there is some s_i along the path, such that *M*, $s_i \models$ *φ2*, and for each *j* < *i*, we have *M*, s_j $\models$ *φ1*

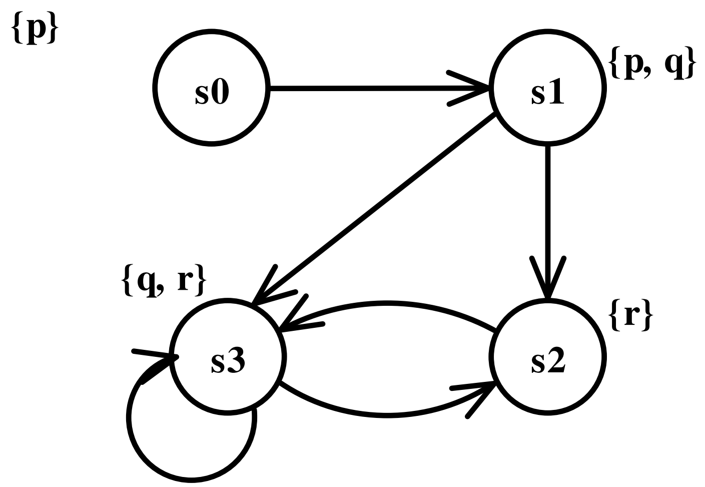




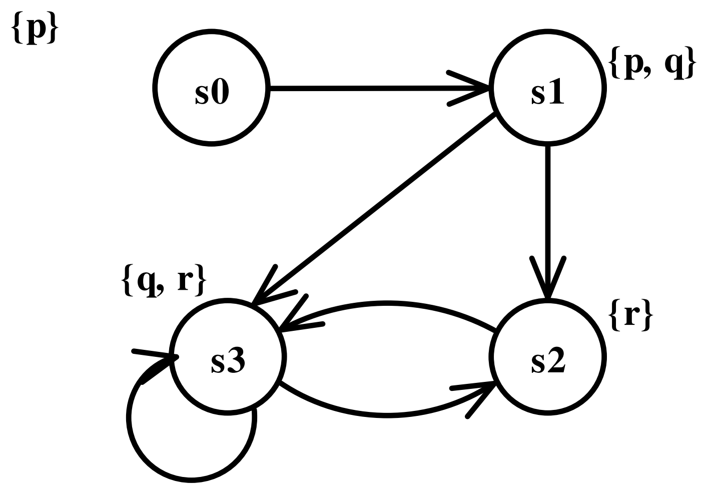
(a) EF g (b) AF g (c) EG g (d) AG g



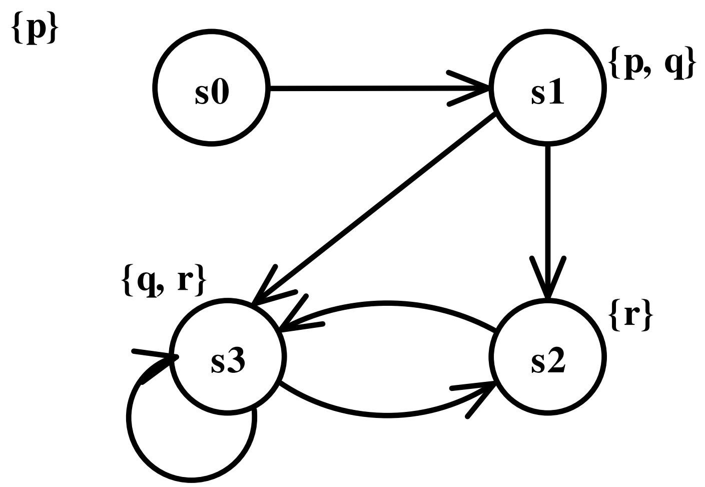
- *S*={s0,s1,s2,s3}
- → ={{s0, s1}, {s1,s2}, {s1, s3}, {s2,s3}, {s3,s2},{s3,s3}}.
- L: L(s0)={p}, L(s1)={p, q}, L(s2)={r}, L(s3)={q, r}.



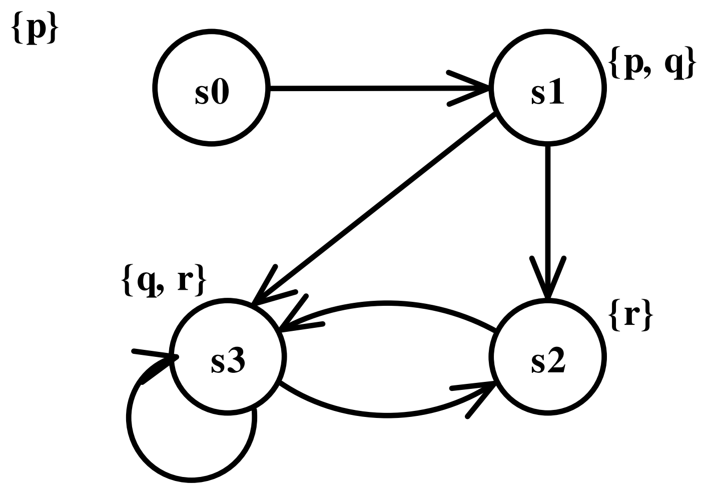
Find the states where the formula AF r holds



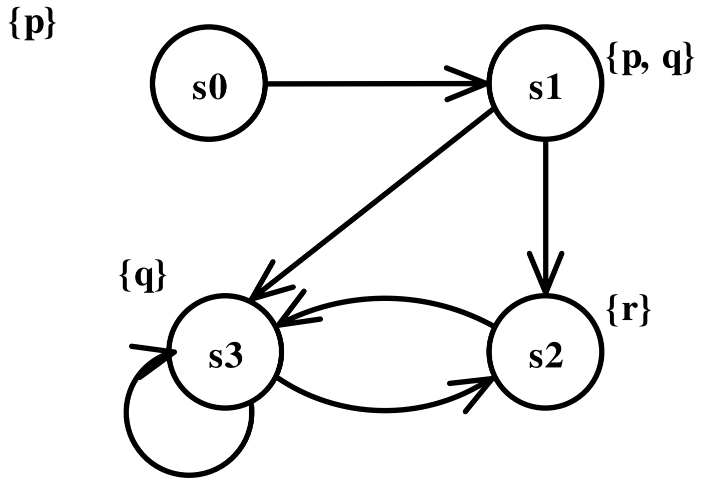
Find the states where the formula AG(AF r) holds



Find the states where the formula $(AF \neg p)$ holds



Find the states where the formula A(p U r) holds



Find the states where the formula (AF r) holds

- In the semantics, the future includes the present.
  - Past, present, future

$M, s \vDash EF\varphi$ holds iff there is one path s1 $\rightarrow$ s2 $\rightarrow$ s3 $\rightarrow$ . . ., where s=s1, and for at least one s_i along the path, $M, s_i \vDash \varphi$.

$M,\ s \models E[\varphi 1\ U\ \varphi 2]$ holds iff for at least one path s1 → s2 → s3 → . . ., where s=s1, $\varphi 1\ U\ \varphi 2$ is satisfied, if there is some s_i along the path, such that $M,\ s_i \models \varphi 2$, and for each $j < i$, we have $M,\ s_j \vdash \varphi 1$

Questions

- Consider X = {p, q, r} be a set of atomic proposition. What is the power set of X.

Questions

Show a Kripke structure such that in a particular state EX (q V r) holds but EX(q □ r) does not hold.

Questions

Show a Kripke structure such that in a particular state AF (q V r) holds but EF(q □ r) does not hold.

Questions

Express the following property in CTL:

It is possible to get a state where started holds, but ready does not hold.

Questions

Express the following property in CTL:

It is possible to get a state where started holds, but ready does not hold.

$$EF(\text{started} \wedge \neg\, \text{ready})$$

Questions

Express the following property in CTL:

For any state, if a request (of some resource) occurs, then it will eventually be acknowledged.

Questions

Express the following property in CTL:

For any state, if a request (of some resource) occurs, then it will eventually be acknowledged.

$$AG(\text{requested} \rightarrow \ AF\ \text{acknowledged})$$

Questions

A certain process is enabled infinitely often on every computation path.

Questions

A certain process is enabled infinitely often on every computation path.

AG (AF enabled)

Questions

From any state it is possible to get a restart state.

Questions

From any state it is possible to get a restart state.

AG (EF restart)

CTL Equivalent formula

CTL Equivalent formula

- Two CTL formulas φ and ψ are said to be semantically equivalent if any state in any model which satisfies one of them also satisfies the other.

NPTEL Phase-II
Video course on

Design Verification and Test of Digital VLSI Designs

Dr. Santosh Biswas
Dr. Jatindra Kumar Deka
IIT Guwahati

Module IV: Temporal Logic

Lecture V: Equivalence between CTL Formulas

Equivalent formula

Propositional Logic

$$p \rightarrow q \equiv \neg p \vee q$$

Predicate Logic

$$\neg \forall x(P(x)) \equiv \exists x(\neg P(x))$$

CTL Equivalent formula

- Two CTL formulas φ and ψ are said to be semantically equivalent if **any state in any model** which satisfies one of them also satisfies the other.

CTL Equivalent formula

- In temporal logic,
  - A : universal quantifier on paths
  - E : existential quantifier on paths
  - G : universal quantifier of states along a path
  - F : existential quantifier of states along a path

$$\neg AF\varphi \equiv EG\neg\varphi$$

$\neg AF\varphi$: “In all paths in future ϕ is true” is false.

$EG\neg\varphi$**:** “There is a path where globally ϕ is not true”

$$\neg AF\varphi \equiv EG\neg\varphi$$

$$\neg EF\varphi \equiv AG\neg\varphi$$

$\neg EF\varphi$: "There is a path where in future ϕ is true" is false

$AG\neg\varphi$ **:** "In all paths globally ϕ is not true"

$$\neg EF\varphi \equiv AG\neg\varphi$$

$$\neg AX\varphi \equiv EX\neg\varphi$$

$\neg AX\varphi$: "In all paths next state satisfies ϕ" is false.

$EX\neg\varphi$ $\equiv$ "There exist a path where in next state ϕ is not true.

$$\neg AX\varphi \equiv EX\neg\varphi$$

$$\neg AF\varphi \equiv EG\neg\varphi$$

$$AF\varphi \equiv \neg EG\neg\varphi$$

$$\neg EF\varphi \equiv AG\neg\varphi$$

$$EF\varphi \equiv \neg AG\neg\varphi$$

$$\neg AX\varphi \equiv EX\neg\varphi$$

$$AX\varphi \equiv \neg EX\neg\varphi$$

$$AF\varphi \equiv A[TU\varphi] \qquad EF\varphi \equiv E[TU\varphi]$$

- AU, EU and EX form an adequate set of temporal operator for CTL.
  - AX can be written with EX
  - AG, EG, AF and EF can be written in terms of AU and EU

Equivalence

EXp	ApUq)	E(pUq)

AXp	≡ ¬EX¬p
AGp	≡ ¬EF¬p
EGp	≡ ¬AF¬p
AFp	≡ A(true U p)
EFp	≡ E(true U p)

$$A[\varphi 1 U \varphi 2] \equiv \neg(E[\neg \varphi 2 U (\neg \varphi 1 \wedge \neg \varphi 2)] \vee EG \neg \varphi 2)$$

$$A[\varphi 1 U \varphi 2] \equiv \neg(E[\neg\varphi 2 U (\neg\varphi 1 \wedge \neg\varphi 2)] \vee EG\neg\varphi 2)$$

$$\neg(E[\neg\varphi 2 U (\neg\varphi 1 \wedge \neg\varphi 2)] \vee EG\neg\varphi 2)$$
$$\equiv \neg E[\neg\varphi 2 U (\neg\varphi 1 \wedge \neg\varphi 2)] \wedge \neg EG\neg\varphi 2$$

$$A[\varphi 1 U \varphi 2] \equiv \neg(E[\neg\varphi 2 U (\neg\varphi 1 \wedge \neg\varphi 2)] \vee EG\neg\varphi 2)$$

$$\neg(E[\neg\varphi 2 U (\neg\varphi 1 \wedge \neg\varphi 2)] \vee EG\neg\varphi 2)$$
$$\equiv \neg E[\neg\varphi 2 U (\neg\varphi 1 \wedge \neg\varphi 2)] \wedge \neg EG\neg\varphi 2$$

$$AF\varphi \equiv \neg EG\neg\varphi$$

$$\neg(E[\neg\varphi 2 U(\neg\varphi 1 \wedge \neg\varphi 2)] \vee EG\neg\varphi 2)$$

$$\equiv \neg E[\neg\varphi 2 U(\neg\varphi 1 \wedge \neg\varphi 2)] \wedge \neg EG\neg\varphi 2$$

$$AF\varphi \equiv \neg EG\neg\varphi$$

Equivalence

EXp	EGp (AFp)	E(pUq)

AXp	$\equiv \neg EX \neg p$
AFp	$\equiv \neg EG \neg p$
AGp	$\equiv \neg EF \neg p$
A(pUq)	$\equiv \neg(EG \neg q \vee E(\neg q \, U \, (\neg p \wedge \neg q)))$
EFp	$\equiv E(\text{true} \, U \, p)$

- Adequate set of temporal operators:
  - AU, EU, EX
  - EG, EU, EX
  - AG, AU, AX
  - AF, EU, EX
  - EG, EU, EX

Other Equivalences

$AG\ p \equiv p \wedge AX\ AG\ p$

$EG\ p \equiv p \wedge EX\ EG\ p$

$AF\ p \equiv p \vee AX\ AF\ p$

$EF\ p \equiv p \vee EX\ EF\ p$

$A[p\ U\ q] \equiv q \vee (p \wedge AX\ A[p\ U\ q])$

$E[p\ U\ q] \equiv q \vee (p \wedge EX\ E[p\ U\ q])$

Other Equivalences

$\mathrm{AG}\ p \equiv p \wedge \mathrm{AX}\ \mathrm{AG}\ p$

Other Equivalences

$\mathrm{EG}\ p \equiv p \land \mathrm{EX}\ \mathrm{EG}\ p$

Other Equivalences

$AF\ p \equiv p \vee AX\ AF\ p$

Other Equivalences

EF p $\equiv$ p $\vee$ EX EF p

Other Equivalences

$A[p \; U \; q] \equiv q \lor (p \land AX \; A[p \; U \; q])$

Other Equivalences

$E[p \; U \; q] \equiv q \vee (p \wedge EX \; E[p \; U \; q])$

Questions

- Which of the following pairs of CTL formulas are equivalent:
  - EFp and EGp
  - $EFp \vee EFq$ and $EF(p \vee q)$
  - $AFp \vee AFq$ and $AF(p \vee q)$
  - $AFp \wedge AFq$ and $AF(p \wedge q)$
  - $EFp \wedge EFq$ and $EF(p \wedge q)$
  - $AG(p \wedge q)$ and $AGp \wedge AGq$
  - T and $AGp \rightarrow EGp$
  - T and $EGp \rightarrow AGP$

Questions

- Which of the following pairs of CTL formulas are equivalent:
  - $EFp \vee EFq$ and $EF(p \vee q)$
  - $AFp \vee AFq$ and $AF(p \vee q)$
  - $AG(p \wedge q)$ and $AGp \wedge AGq$
  - T and $AGp \rightarrow EGp$

Questions

- Which of the following pairs of CTL formulas are equivalent:
  - EFp and EGp
  - $AFp \wedge AFq$ and $AF(p \wedge q)$
  - $EFp \wedge EFq$ and $EF(p \wedge q)$
  - T and $EGp \rightarrow AGP$

Questions

- Consider the formula

  $E(Fp \wedge Fq)$

Questions

- Consider the formula

  $E(Fp \wedge Fq)$: not a CTL formula

If we have $Fp \wedge Fq$ along any path, then either p must come before q, or the other way round.

Questions

- Consider the formula

  $E(Fp \wedge Fq)$: not a CTL formula

If we have $Fp \wedge Fq$ along any path, then either p must come before q, or the other way round.

$EF(p \wedge EFq) \vee EF(q \wedge EFP)$

Questions

- Consider the formula

  $E(Fp \wedge Fq)$

  $EF(p \wedge EFq) \vee EF(q \wedge EFP)$