Model Checking with CTL

Presented by Jason Simas

Model Checking with CTL

Based Upon:

Logic in Computer Science. Huth and Ryan. 2000. (148-215)

Model Checking. Clarke, Grumberg and Peled. 1999. (1-26)

Content

- Context
  - Model Checking
  - Models
- CTL
  - Syntax
  - Semantics
  - Checking Algorithm

Model Checking

- $M \models \varphi$
  - M is the model
    - Requires a description language
  - φ is the property to check
    - Requires a specification language
  - $\models$ is the "satisfaction relation"
    - Algorithm to check whether $(M, \varphi)\ \varepsilon \models$
    - Outputs either "yes" or "no" (+ trace)

Models

- Fundamentals
- Language Definition
- Example Model

Fundamentals

- Want to prove properties
- Model all relevant sub properties
- Model abstraction level <= properties
- -> Model how properties change
  - Over time? (sort of)
  - Over property change? (yes)
    - Abstract out time

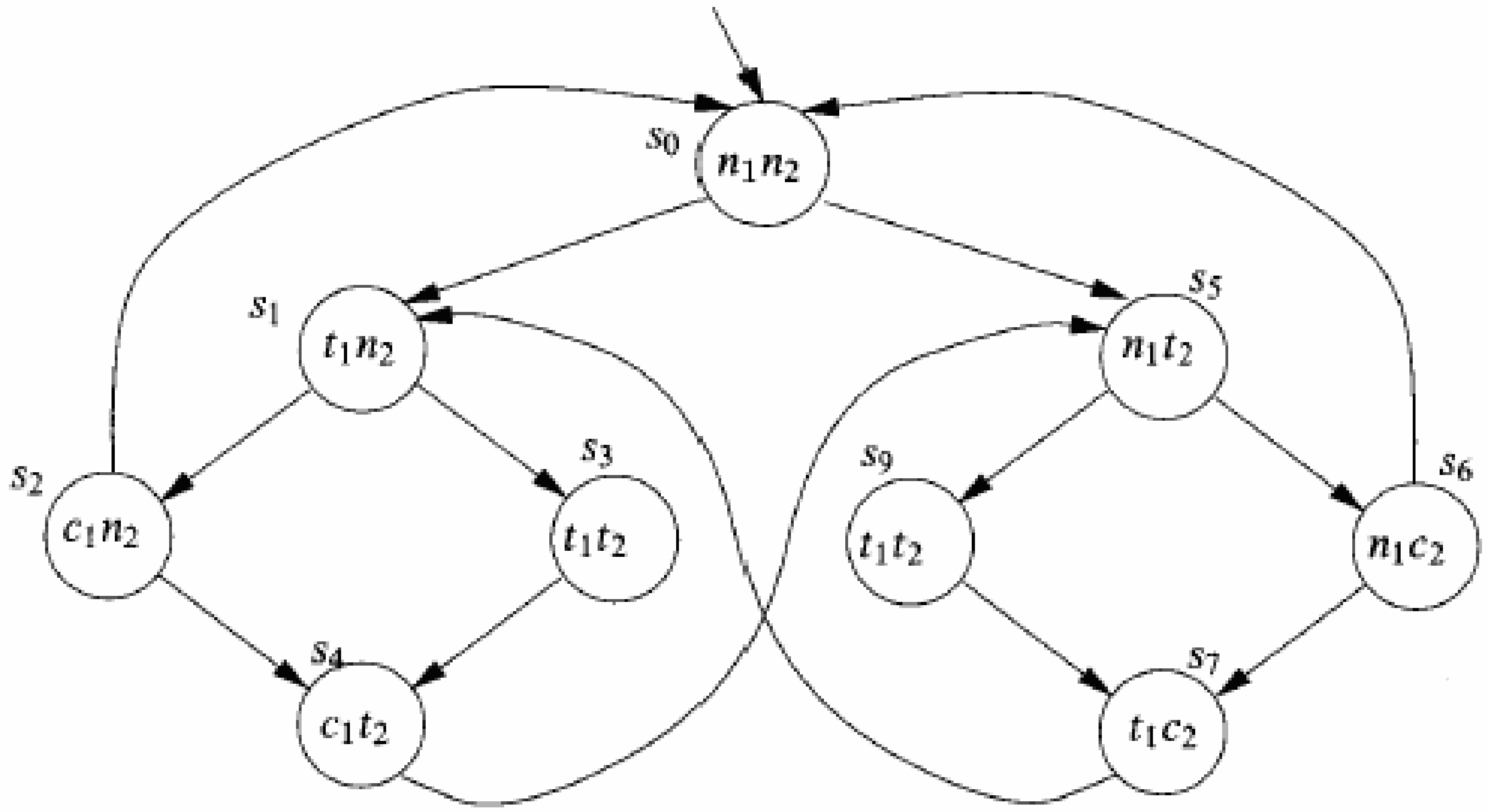
Modeling Property Change

- Model = States + Transitions + Labels
  - States
    - Possibilities of which properties can be true together +
    - Possibilities of which properties can follow each other +
  - Transitions
    - Possibilities of which states can follow each other
  - Labels
    - Possibilities of which properties are true for each state
- States need not be unique wrt labels
- Use a directed graph

Definition: Model for CTL

- $M = (S, \to, L)$
  - S is a finite set of states $\{s_0, s_1, \ldots, s_n\}$
  - $\to$ is a set of transitions
    - $\to \subseteq SxS$ and
    - for every $s \in S$ there is some $s' \in S$ such that $s \to s'$
  - L is a labeling function $L: S \to \mathcal{P}$ (Atoms)
    - S is the set of states of $\mathcal{M}$
    - $\mathcal{P}$ (Atoms) is the power set of Atoms
      - Atoms is the set of all propositions

Mutual Exclusion (Interleaved)



Mutual Exclusion (Interleaved)

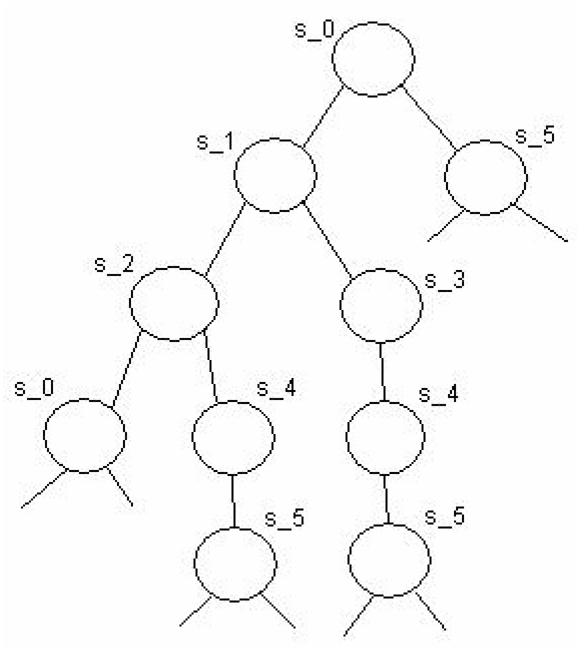
- $M = (S, \rightarrow, L)$ where
  - $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_9,\}$
  - $\rightarrow$ =
    - $\{(s_0, s_1), (s_1, s_2), (s_1, s_3), (s_2, s_4), (s_3, s_4), (s_2, s_0), (s_4, s_5),$
    - $(s_0, s_5), (s_5, s_6), (s_5, s_9), (s_6, s_7), (s_9, s_7), (s_6, s_0), (s_7, s_1)\}$
  - L =
    - $\{(s_0, \{n_1, n_2\}),$
    - $(s_1, \{ t_1, n_2\}), (s_2, \{ c_1, n_2\}), (s_3, \{ t_1, t_2\}), (s_4, \{c_1, t_2\}),$
    - $(s_5, \{ n_1, t_2\}), (s_6, \{ n_1, c_2\}), (s_7, \{t_1, c_2\}), (s_9, \{ t_1, t_2\})\}$
- Note: s_3, s_9 are distinct for "turns"

Properties

- Remember M |= φ
  - φ specifies properties of states/transitions
  - Need a specification language for φ, CTL
- CTL: Computation Tree Logic
  - Specifying properties of "computation trees"
  - "Logic" = Language + Inference Rules
    - Inference Rules = Algorithm for check

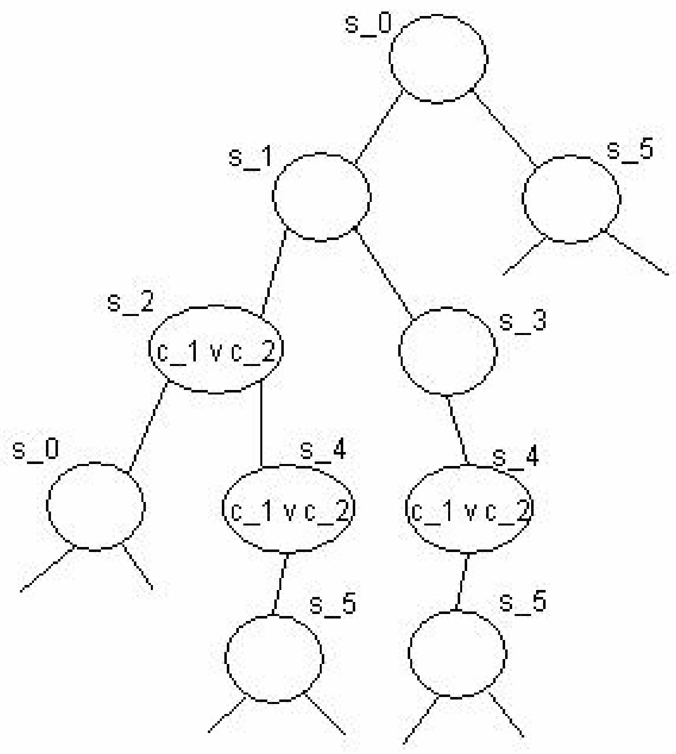
Computation Trees

- A tree such that starting at some state s,
  - There exists edges to each of its children ($s \to s'$)
  - Same is true for each child, ad infinitum



Example: “Efficiency”

- For each “cycle” ($n_i \to^* n_i$) some process enters its critical section
- CTL: AG $((s_1 \vee s_5) \to AX\ (A\neg(s_1 \vee s_5)\ U\ (c_1 \vee c_2)))$



CTL Syntax

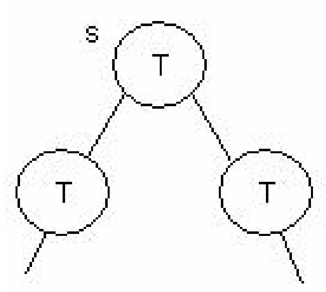
- $\varphi :=$
  - $\perp \mid \top \mid p \mid$
  - $(\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid$
  - $AX\ \varphi \mid EX\ \varphi \mid A[\varphi\ U\ \varphi] \mid E[\varphi\ U\ \varphi] \mid AG\ \varphi \mid EG\ \varphi \mid AF\ \varphi \mid EF\ \varphi$
- Atoms: $\perp$, $\top$, p
  - p is an arbitrary atomic property either true or false
    - Example: c_1: "process 1 is in its critical section"
- Propositional Connectives: $\wedge$, $\vee$, $\neg$, $\rightarrow$
- Temporal Connectives: EG, AG, EX, AX, EF, AF, EU, AU
  - Note: $EU\ \varphi_1\ \varphi_2$ same as $E\ [\varphi_1\ U\ \varphi_2]$
- Binding Precedence:
  - Unary Connectives: $\neg$, AX, EX, AG, EG, AF, EF
  - Binary Connectives: $\rightarrow$, AU, EU
- $\top$, c_1, $c_1 \wedge c_2$, $AX\ (c_1 \wedge c_2)$, $A\ [c_1\ U\ c_1]$, $E\ [\top\ U\ (AX\ (c_1 \wedge c_2))]$

CTL Semantics

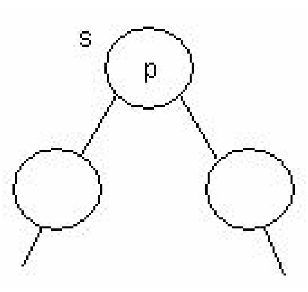
- M,s |= φ where φ is a CTL formula
  - "is φ true for the model M at state s?"
  - when s is the initial state: M |= φ
  - Irrelevant whether φ is true/false at other states
- Temporal Connectives:
  - A,E: range over paths from s
  - G,X,F,U: range over states on a path from s

$\top$, $\bot$, p

- M,s |= $\top$ and not M,s |= $\bot$ for all s ε S

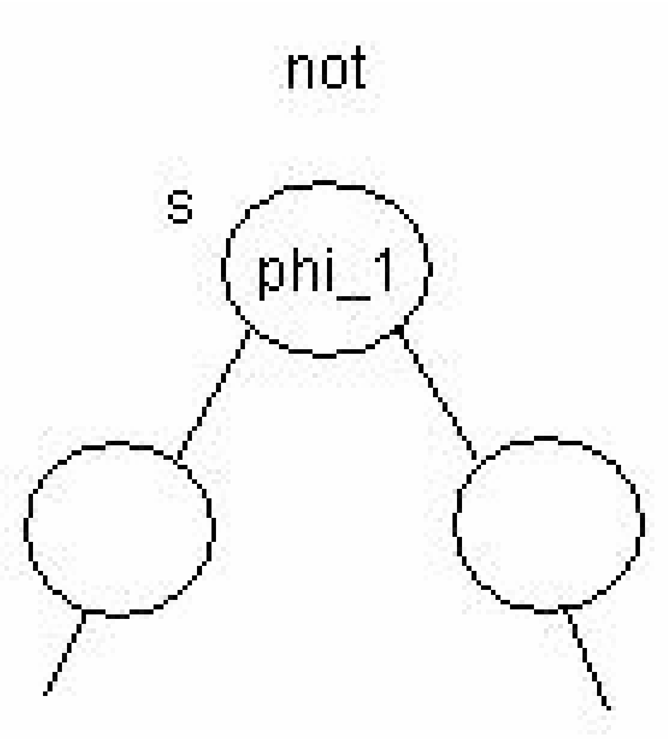


- M,s |= p iff p ε L(s)



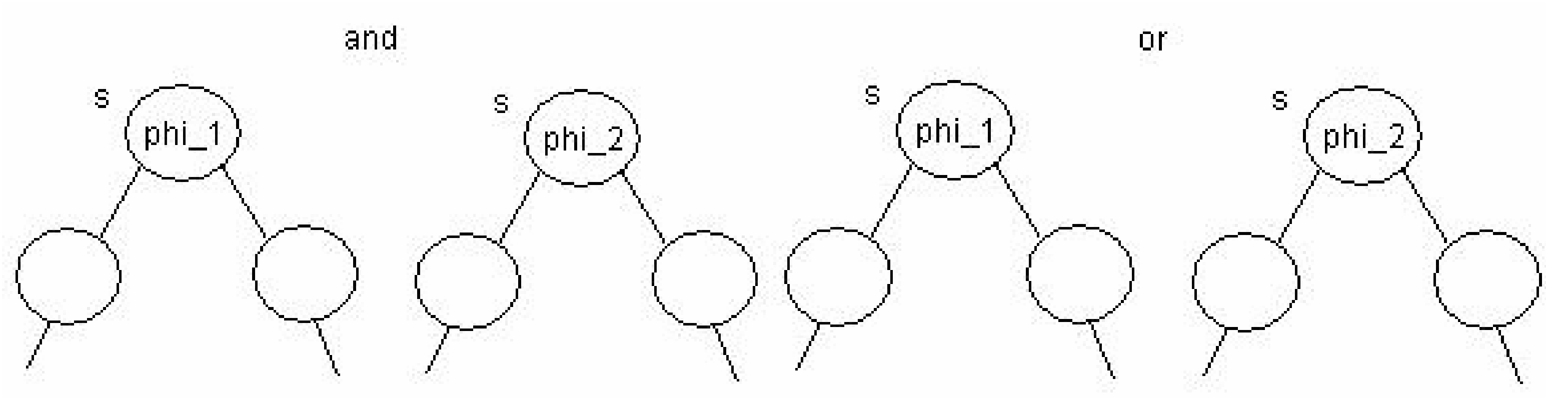
¬

- M,s |= ¬φ iff not M,s |= φ



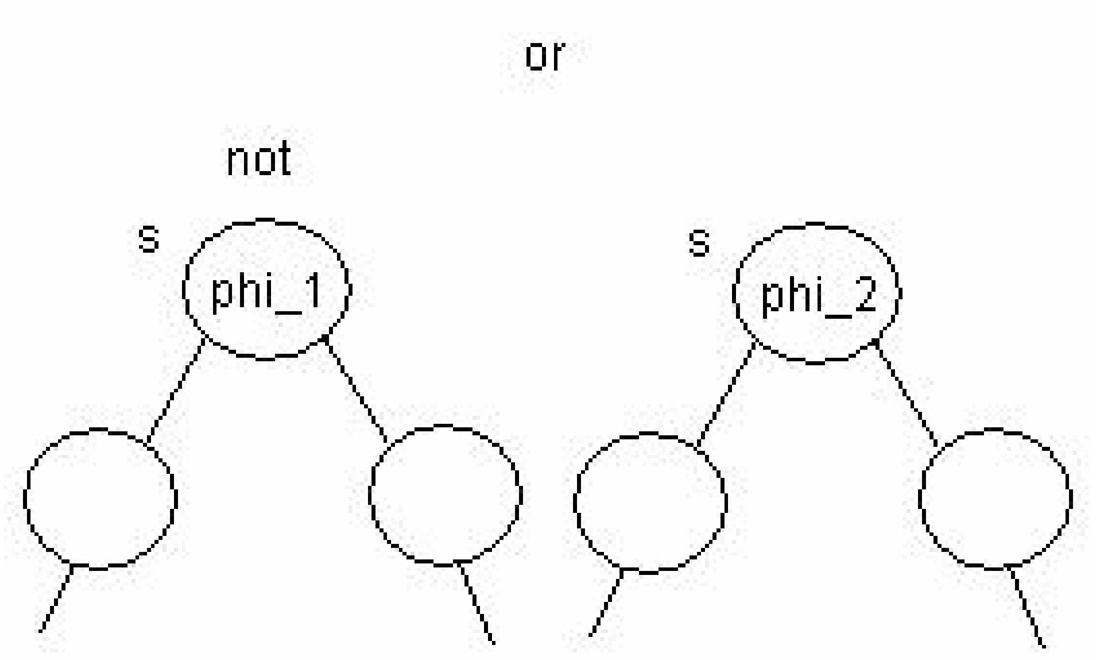
^, v

- $M,s \models \varphi_1 \wedge \varphi_2$ iff $M,s \models \varphi_1$ and $M,s \models \varphi_2$
- $M,s \models \varphi_1 \vee \varphi_2$ iff $M,s \models \varphi_1$ or $M,s \models \varphi_2$



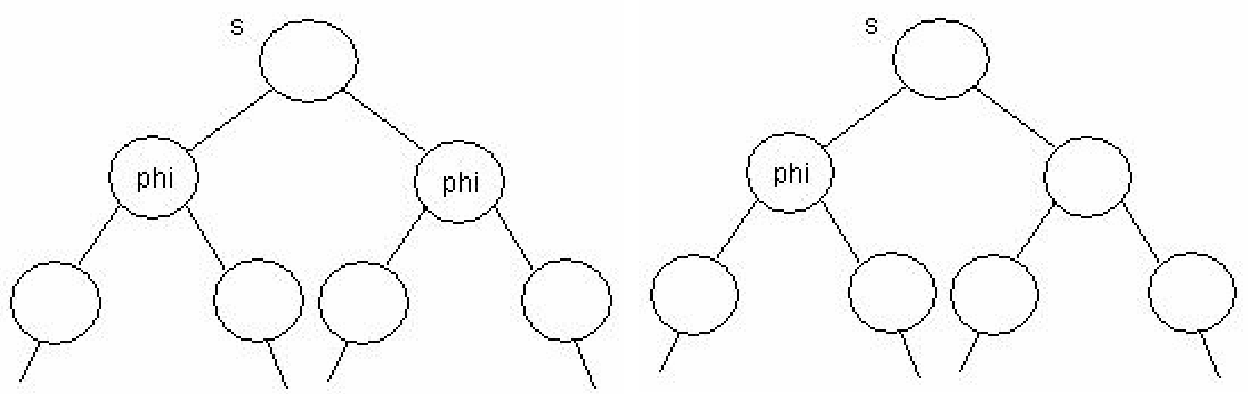
$\rightarrow$

- $M,s \models \varphi_1 \rightarrow \varphi_2$ iff not $M,s \models \varphi_1$ or $M,s \models \varphi_2$



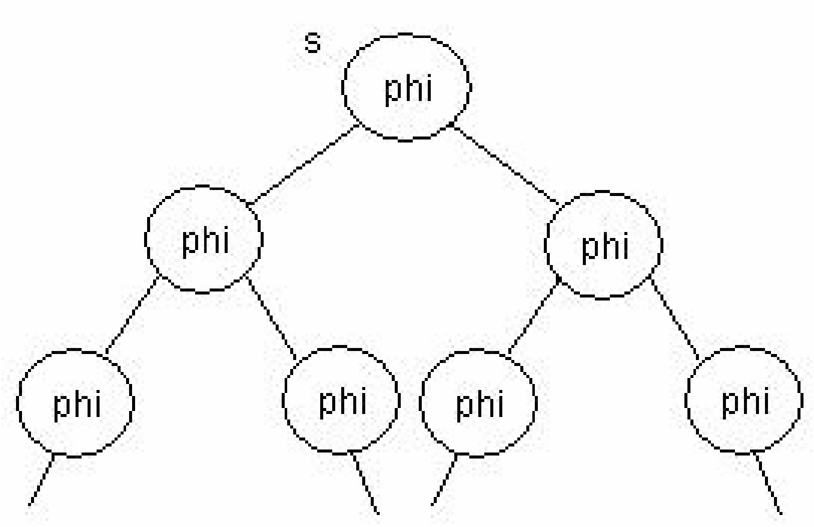
AX, EX

- AX
  - M,s |= AX φ iff for all s′ such that s→s′ we have M,s′ |= φ
  - "For all paths, for the next state, φ is true"
- EX
  - M,s |= EX φ iff for some s′ such that s→s′ we have M,s′ |= φ
  - "For some path, for the next state, φ is true"



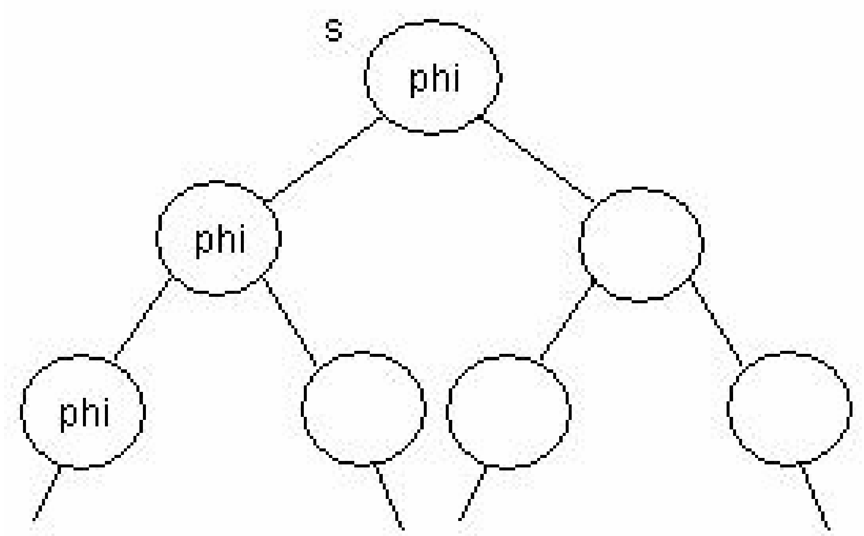
AG

- M,s |= AG φ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s and for all s_i along the path, we have M,s_i |= φ
  - "For all paths, for all states along each path, φ is true"



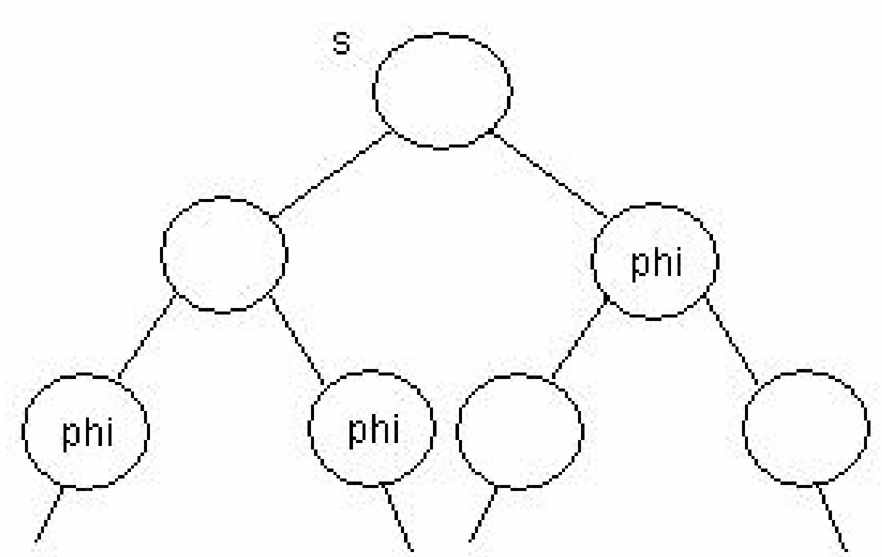
EG

- $M,s \models EG\ \varphi$ iff for some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s and for all s_i along the path, we have $M,s_i \models \varphi$
  - "For some path, for all states along the path, φ is true"



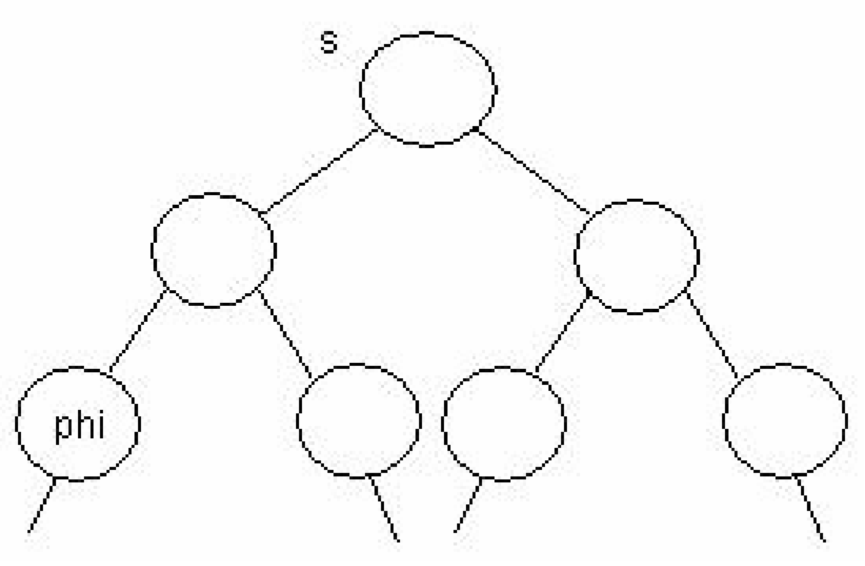
AF

- $M,s \models AF\ \varphi$ iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 is s and for some s_i along each path, we have $M,s_i \models \varphi$
  - "For all paths, for some state along each path, φ is true"



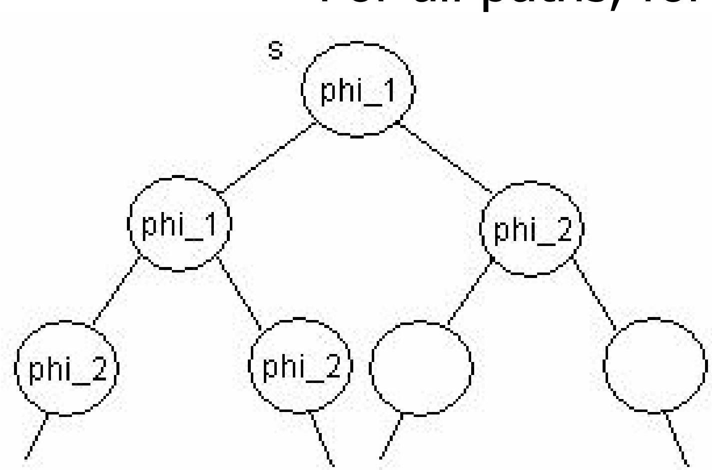
EF

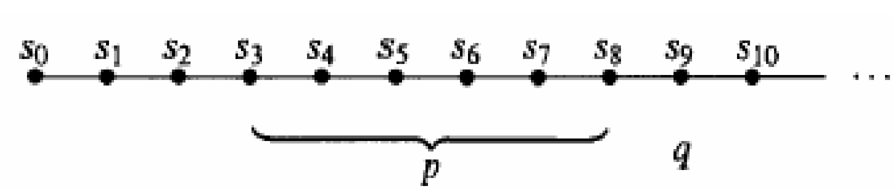
- $M,s \models EF\ \varphi$ iff for some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s and for some s_i along the path, we have $M,s_i \models \varphi$
  - "For some path, for some state along each path, φ is true"



AU

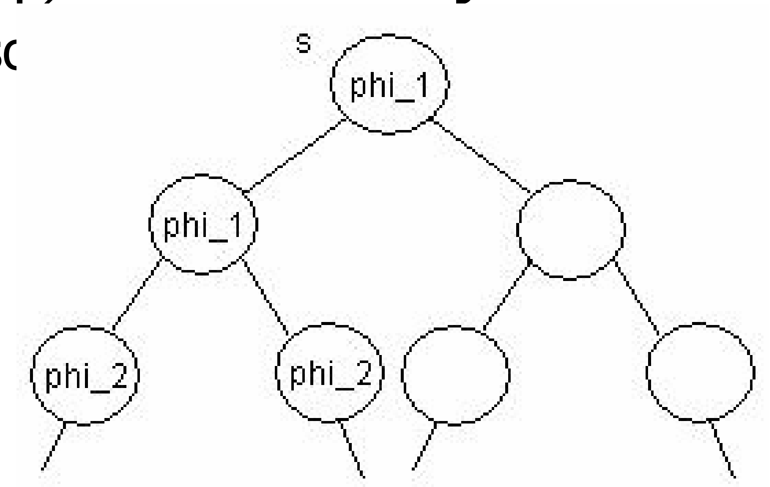
- M,s |= A [φ_1 U φ_2] iff for all paths $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$ where s_1 equals s, each path satisfies φ_1 U φ_2 (i.e. there is some s_i along the path such that M,s_i |= φ_2 and for each $j < i$ we have M,s_j |= φ_1)
  - "For all paths, for every state in each path φ_1 until φ_2"





EU

- $M,s \models E[\varphi_1 \; U \; \varphi_2]$ iff for some path $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots$
  where s_1 equals s, the path satisfies $\varphi_1 \; U \; \varphi_2$ (i.e. there is some s_i along the path such that $M,s_i \models \varphi_2$ and for each $j < i$ we have $M,s_j \models \varphi_1$)
  - "For sc path φ_1 until φ_2"



Inclusion of "s" in Condition

- "s" is the first state checked
  - For G, F, U
  - But not for X
- Examples:
  - $M \models AF\ (n_1 \wedge n_2)$
  - $M \models EG\ \neg(n_1 \wedge n_2)$
  - $M \models A\ [\perp U\ (n_1 \wedge n_2)$
- To exclude "s", use $X\ \varphi$

Mutual Exclusion Properties

- Safety:
  - Only one process shall be in its critical section at any time
  - $AG\ \neg(c_1 \wedge c_2)$
- Liveness:
  - Whenever any process wants to enter its critical section, it will eventually be permitted to do so
  - $AG\ (t_1 \rightarrow AF\ c1) \wedge AG\ (t_2 \rightarrow AF\ c_2)$
- Non-blocking
  - A process can always request to enter its critical section
  - $AG\ (n_1 \rightarrow EX\ t_1) \wedge AG\ (n_2 \rightarrow EX\ t_2)$
- No strict sequencing:
  - Processes need not enter their critical section in strict sequence
  - $EF\ (c_1 \wedge E[c_1\ U\ (\neg c_1 \wedge E[\neg c_2\ U\ c_1])]) \vee$
    $EF\ (c_2 \wedge E[c_2\ U\ (\neg c_2 \wedge E[\neg c_1\ U\ c_2])])$

Checking Algorithm

- Minimal Set of Connectives
- Algorithm
- Correctness
- Complexity
- Implementation

Minimal Set of Connectives

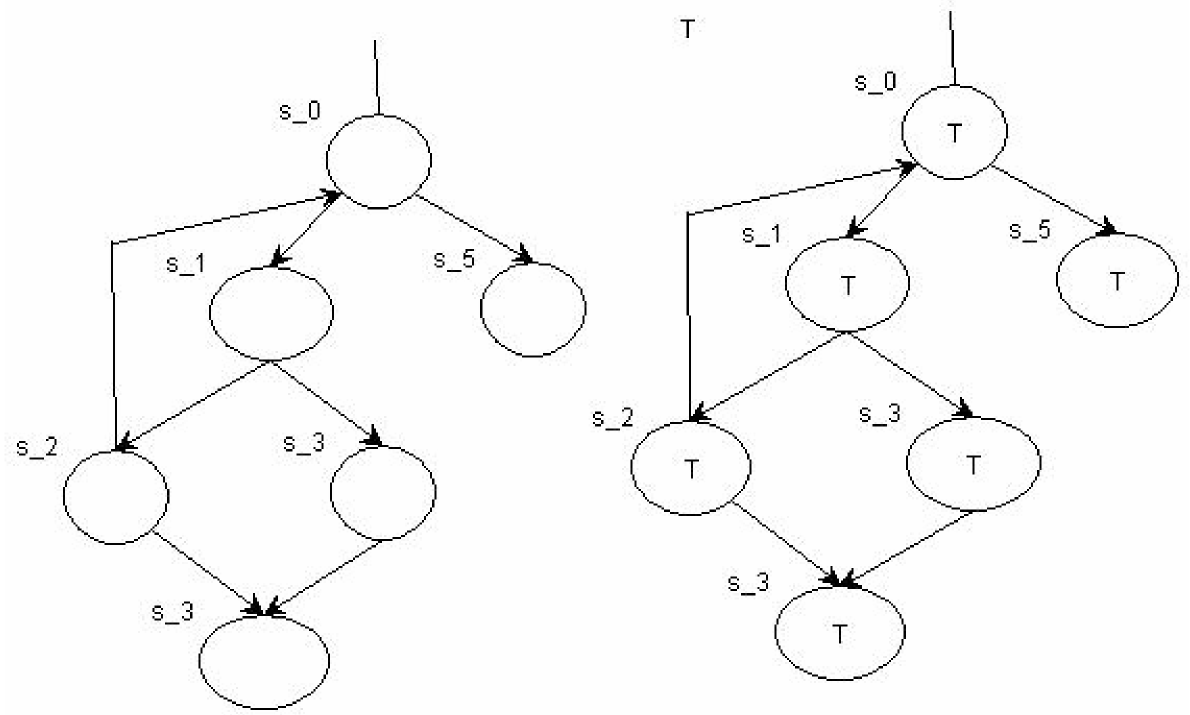
- Two CTL formulas φ and ψ are semantically equivalent iff any state in any model which satisfies one of them also satisfies the other
  - De Morgan's Law
    - $\neg AF\ \varphi = EG\ \neg\varphi$
    - $\neg EF\ \varphi = AG\ \neg\varphi$
  - Minimal Set of Connectives: $\wedge$, $\neg$, $\bot$, AF, EX, EU
    - Translate: AG, EG, EF, AX, AU
    - For AG: $AG\ \varphi = \neg EF\ \neg\varphi$
    - For EG: $EG\ \varphi = \neg AF\ \neg\varphi$
    - For EF: $EF\ \varphi = E\ [\top\ U\ \varphi]$
    - For AX: $AX\ \varphi = \neg EX\ \neg\varphi$
    - For AU: $A\ [\top\ U\ \varphi] = AF\ \varphi$

Algorithm

- Input: The model M and the CTL formula φ
- Output: The set of states of M that satisfy φ
- Steps:
  - Translate φ to φ′ where φ′ only has connectives in the minimal set
  - Label the states of M with the sub formulas of φ that are satisfied there, starting with the smallest sub formulas and working outwards towards φ
- If s_0 is an element of the output, then "yes"

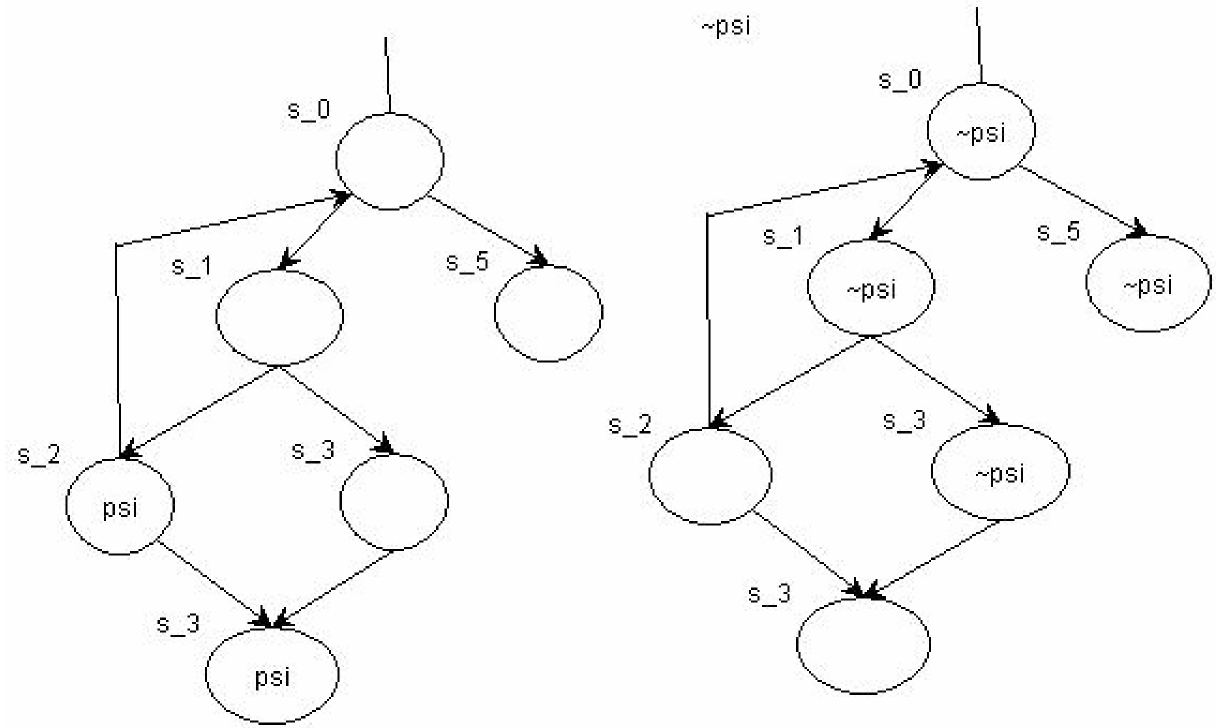
$\top$, $\bot$

- $\bot$: then no states are labeled with $\bot$
- $\top$: then all states are labeled with $\top$



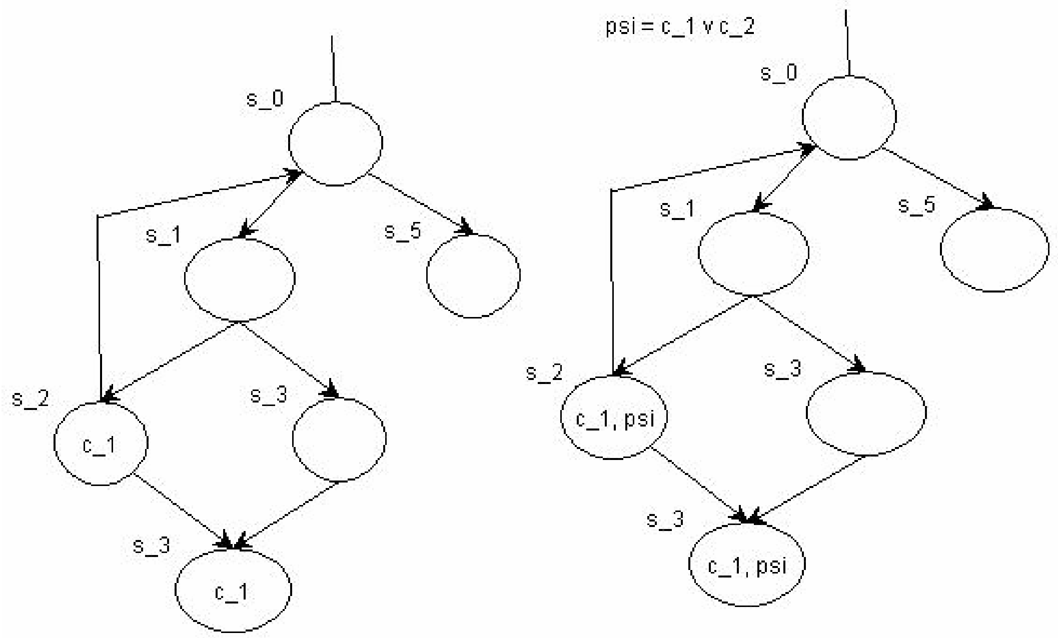
P, ¬

- p: then label s with p if p ε L(s)
- $\neg\psi_1$: label s with $\neg\psi_1$ if s is not already labeled with ψ_1



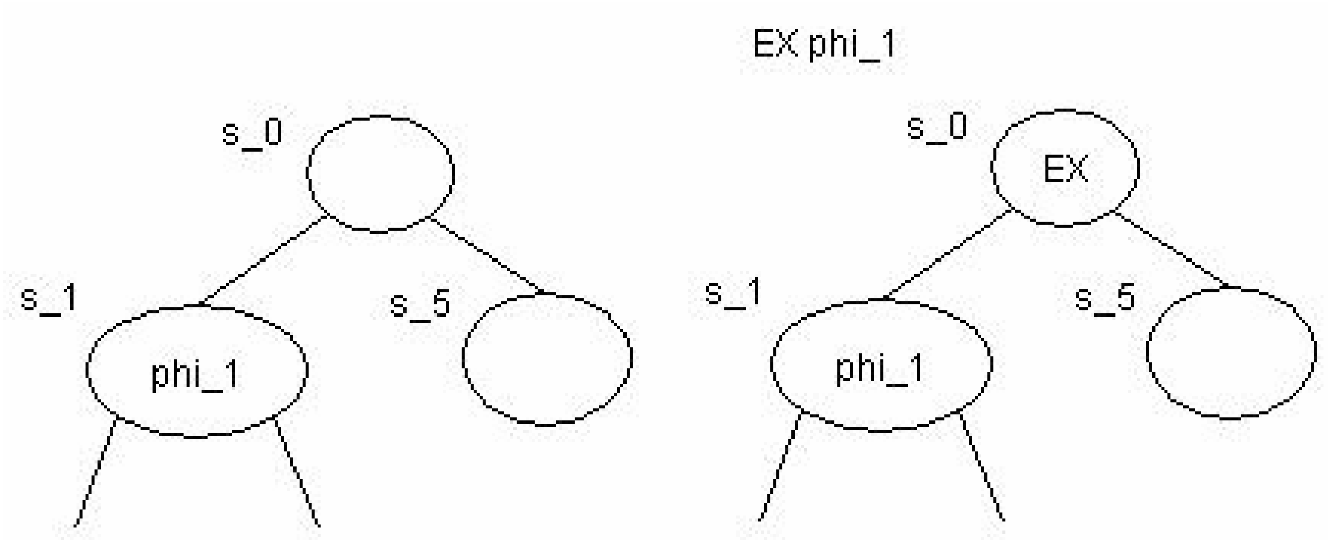
^, v

- ψ_1 ^ ψ_2: label s with ψ_1 ^ ψ_2
  if s is already labeled with both ψ_1 and ψ_2
- ψ_1 v ψ_2: label s with ψ_1 v ψ_2
  if s is already labeled with ψ_1 or ψ_2



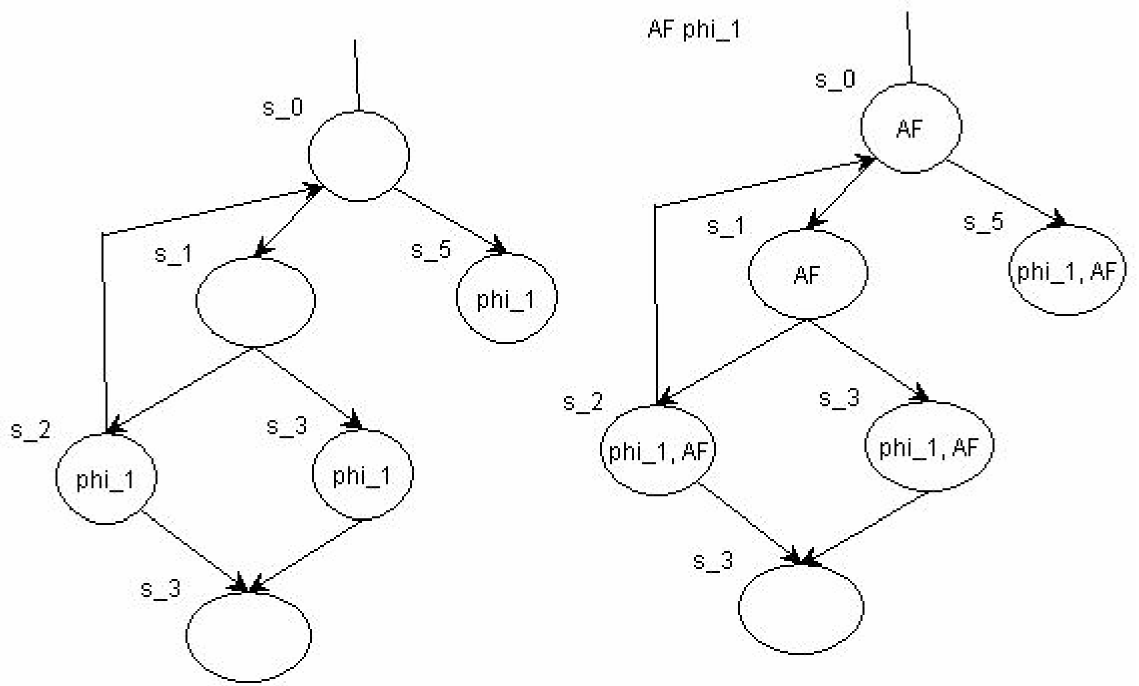
EX

- EX ψ_1: label any state with EX ψ_1 if one of its successors is labeled with ψ_1



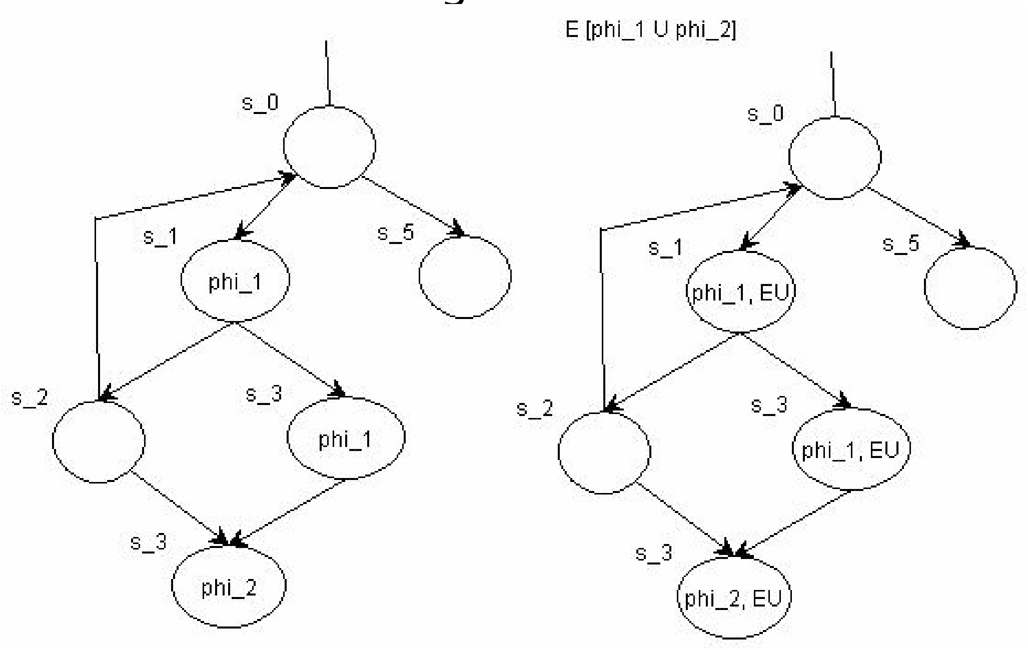
AF

- AF ψ_1:
  - If any state s is labeled with ψ_1, label it with AF ψ_1
  - Repeat: label any state with AF ψ_1 if all successor states



EU

- $E[\psi_1 \ U \ \psi_2]$:
  - If any state s is labeled with ψ_2, label it with $E[\psi_1 \ U \ \psi_2]$
  - Repeat: label any state with $E[\psi_1 \ U \ \psi_2]$ if it is labeled with ψ_1 and at least one of its successors is labeled with $E[\psi_1 \ U \ \psi_2]$, until there is no change



Correctness: Termination

- Repeat until no change of AF and EU
  - Required since algorithm may add states and existence of states is part of condition
- Problem: "repeat" may not terminate
- Show that the functions for AF and UE terminate
  - Show that
    $F_0\ (F_1\ (\ldots\ F_n\ (S))) = F_0\ (F_1\ (\ldots\ F_{n+1}\ (S)))$ for some n

Fixpoints

- Given: F is a function F: P(S) → P(S)
- Fixpoint Sets
  - A subset X of S is called a fixpoint of F if F(X) = X
  - If we prove "repeat" has a fixpoint
    then we've proved "repeat" terminates
- Known Theorem:
  Every monotone function has a fixpoint
  - Is "repeat" monotone?

Monotone Functions

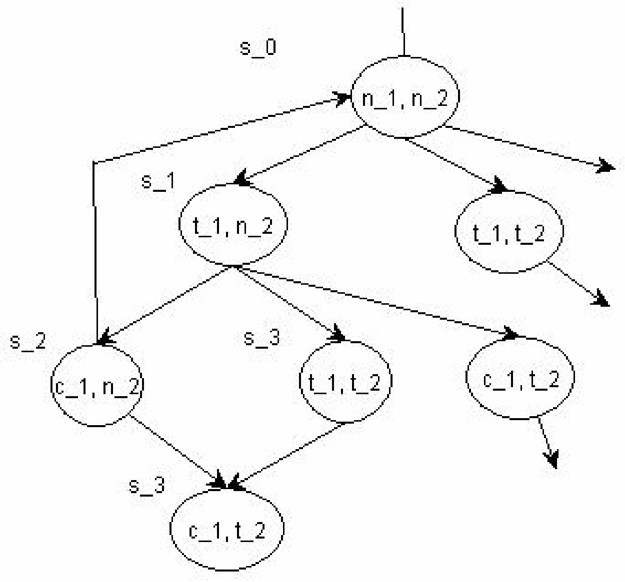
- Monotone Functions:
  - F is monotone
    iff $X \subseteq Y$ implies $F(X) \subseteq F(Y)$ for all subsets X and Y of S
- F_{AF} is monotone
  - X,Y are the set of states with a label $AF\varphi$
  - F_{AF} only adds states, that is $F_{AF}(Z) = Z \cup \{...\}$
  - Condition for what is in {…} is dependent on Z
  - "More states in Z, then more potential for adding states"
  - Since X is "contained" in Y, then Y has all the potential of X
    (i.e. $F_{AF}(X) = F_{AF}(Y)$)
  - And if X is smaller than Y, then Y has more potential than X
    (i.e. $F_{AF}(X) \subseteq F_{AF}(Y)$)
  - So if $X \subseteq Y$ then $F_{AF}(X) \subseteq F_{AF}(Y)$
- F_{EU} is similarly monotone

Complexity

- This Algorithm: O (f * V * (V + E))
  - f is the number of connectives in the formula
  - V is the number of states
  - E is the number of transitions
  - "linear in the size of the formula and quadratic in the size of the model"
- Better Algorithms: O (f * (V + E)

Complexity: State Explosion

- Problem is size of model, not algorithm
  - Size of model (V + E) is exponential
    in the number of variables (or properties on them)
  - Size of model (V+E) is exponential
    in the number of components that can execute in parallel



Implementations

- SMV
  - Model Checker
  - Available from CMU
  - Created by K. McMillan
- NuSMV
  - Reimplementation
- Cadence SMV
  - Reimplementation + Compositional Focus