Design Verification

Model/Property Checking: Computation Tree Logic

Virendra Singh

Associate Professor

Computer Architecture and Dependable Systems Lab

Department of Electrical Engineering

Indian Institute of Technology Bombay

http://www.ee.iitb.ac.in/~viren/

E-mail: viren@ee.iitb.ac.in

EE-709: Testing & Verification of VLSI Circuits

Lecture 34 (09 April 2013)

CADSL

Semantics of CTL

Let $M = (S, \rightarrow, L)$ be a model for CTL, and s in S, φ a CTL formula. The relation $M,s \models \varphi$ is defined by structural induction on ϕ.

$$1. M, s \models T$$

$$2. M, s \not\models \underline{I}$$

$$3. M, s \models p, iff, p \in L(s_1)$$

$$4. M, s \models \neg\phi, iff, M, s \not\models \phi$$

$$5. M, s \models \phi_1 \wedge \phi_2, iff, M, s \models \phi_1 and, M, s \models \phi_2$$

$$6. M, s \models \phi_1 \vee \phi_2, iff, M, s \models \phi_1 or, M, s \models \phi_2$$

$$7. M, s \models \phi_1 \rightarrow \phi_2, iff, M, s \models \phi_2 whenever, M, s \models \phi_1$$

Semantics of CTL

$8. M, s \models AX\phi, iff, \forall s_1, s.t., s \rightarrow s_1; M, s_1 \models \phi$

$9. M, s \models EX\phi, iff, \exists s_1, s.t., s \rightarrow s_1; M, s_1 \models \phi$

$10. M, s \models AG\phi, iff, \forall paths, s_1 \rightarrow s_2 \rightarrow s_3 \ldots, \forall s_i, M, s_i \models \phi$

$11. M, s \models EG\phi, iff, \exists path, s_1 \rightarrow s_2 \rightarrow s_3 \ldots, \forall s_i; M, s_i \models \phi$

$12. M, s \models AF\phi, iff, \forall paths, s_1 \rightarrow s_2 \rightarrow s_3 \ldots, \exists s_i; M, s_i \models \phi$

$13. M, s \models EF\phi, iff, \exists path, s_1 \rightarrow s_2 \rightarrow s_3 \ldots, \exists s_i; M, s_i \models \phi$

$14. M, s \models A[\phi U \varphi], iff, \forall paths, s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots, \forall s_i,$

$M, s_i \models \phi_2, \forall j < i; M, s_j \models \phi_1$

$15. M, s \models E[\phi U \varphi], iff, \exists path, s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \ldots, \forall s_i,$

$M, s_i \models \phi_2, \forall j < i; M, s_j \models \phi_1$

Computation Tree Logic - Equivalence

$$AX\phi \equiv \neg EX \neg \phi$$

$$AF\phi \equiv A[TU\phi]$$

$$EF\phi \equiv E[TU\phi]$$

$$AG\phi \equiv \neg EF \neg \phi \equiv \neg E[TU \neg \phi]$$

$$EG\phi \equiv \neg AF \neg \phi \equiv \neg A[TU \neg \phi]$$

Essential Set: AU, EU, and EX

Adequate Set of CTL Operators

Theorem: A set of temporal connectives in CTL is adequate if, and only if, it contains at least one of {AX, EX}, at least one of {EG, AF, AU} and EU.

Checking CTL Formula

1. find all nodes at which the formula holds
2. determines whether all initial states are contained in the set of nodes

Kripke structure

- ❖ Labeled variables are true, and the missing variables are false
- ❖ extend this labeling rule to include formulas or subformulas that evaluate true at the node

Checking CTL Formula

- consider AND ($\wedge$) and NOT ($\neg$) operators
- if both operand formulas are true at the node, the resulting AND formula is true at the node and it is labeled at the node
- If the operand formula is not true (in other words, it is missing at a node), then the resulting NOT formula is true and it is labeled at the node

- only need to consider EXΦ, E(ΦUΨ), and AFΦ temporal operators

Checking CTL Formula

- Algorithm for Checking AF(Φ)

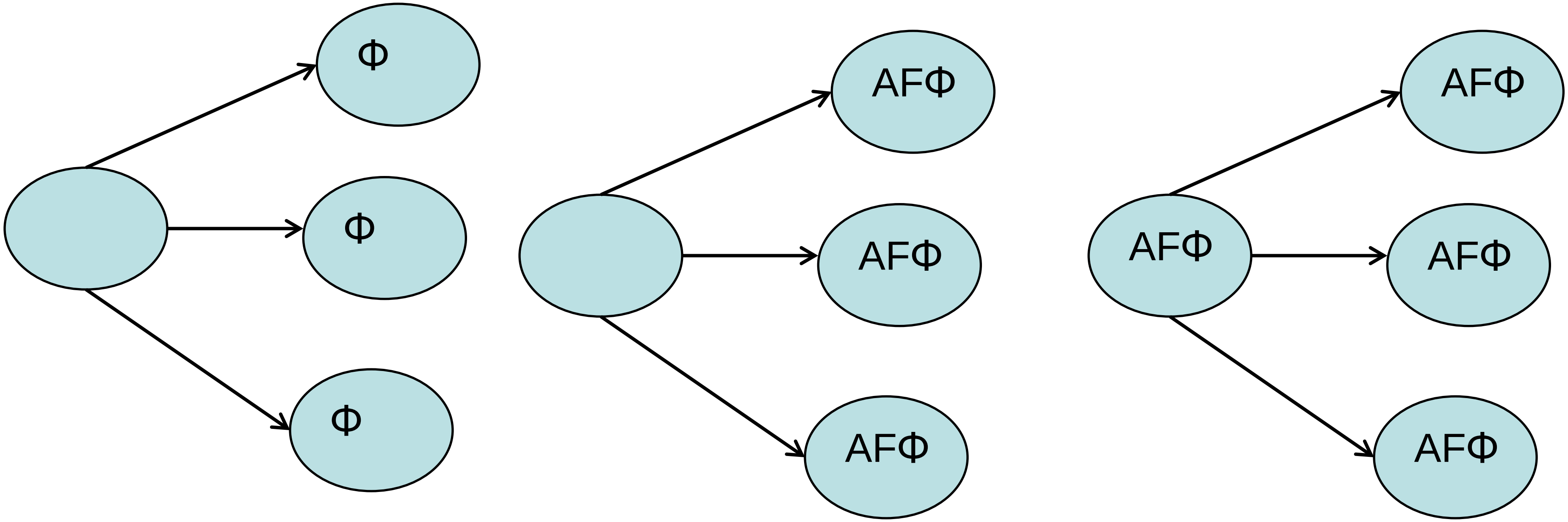
input: a Kripke structure K and a CTL formula EX(Φ).

output: labeling of the states where AF(Φ) holds

Verify_AF(Φ): // check CTL formula AF(Φ)

for each state s of K, add label AF(Φ) if Φ is labeled at a all successor of s

Checking CTL Formula: AFΦ



Checking CTL Formula

- Algorithm for Checking EX(Φ)

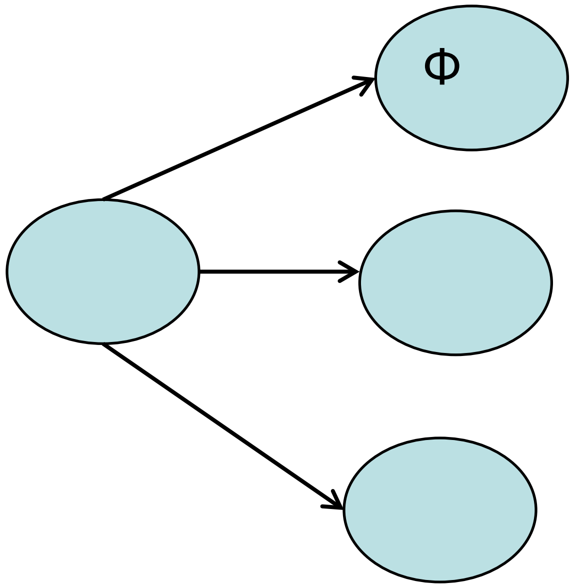
input: a Kripke structure K and a CTL formula EX(Φ).

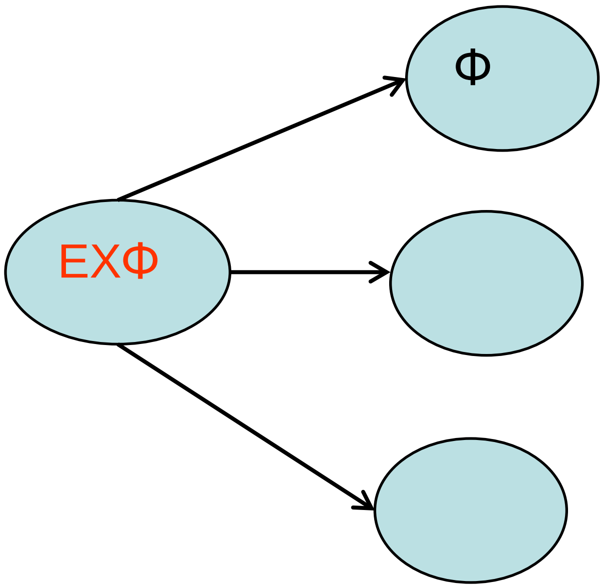
output: labeling of the states where EX(Φ) holds

Verify_EX(Φ): // check CTL formula EX(Φ)

for each state s of K, add label EX(Φ) if Φ is labeled at a successor of s

Checking CTL Formula: EXΦ



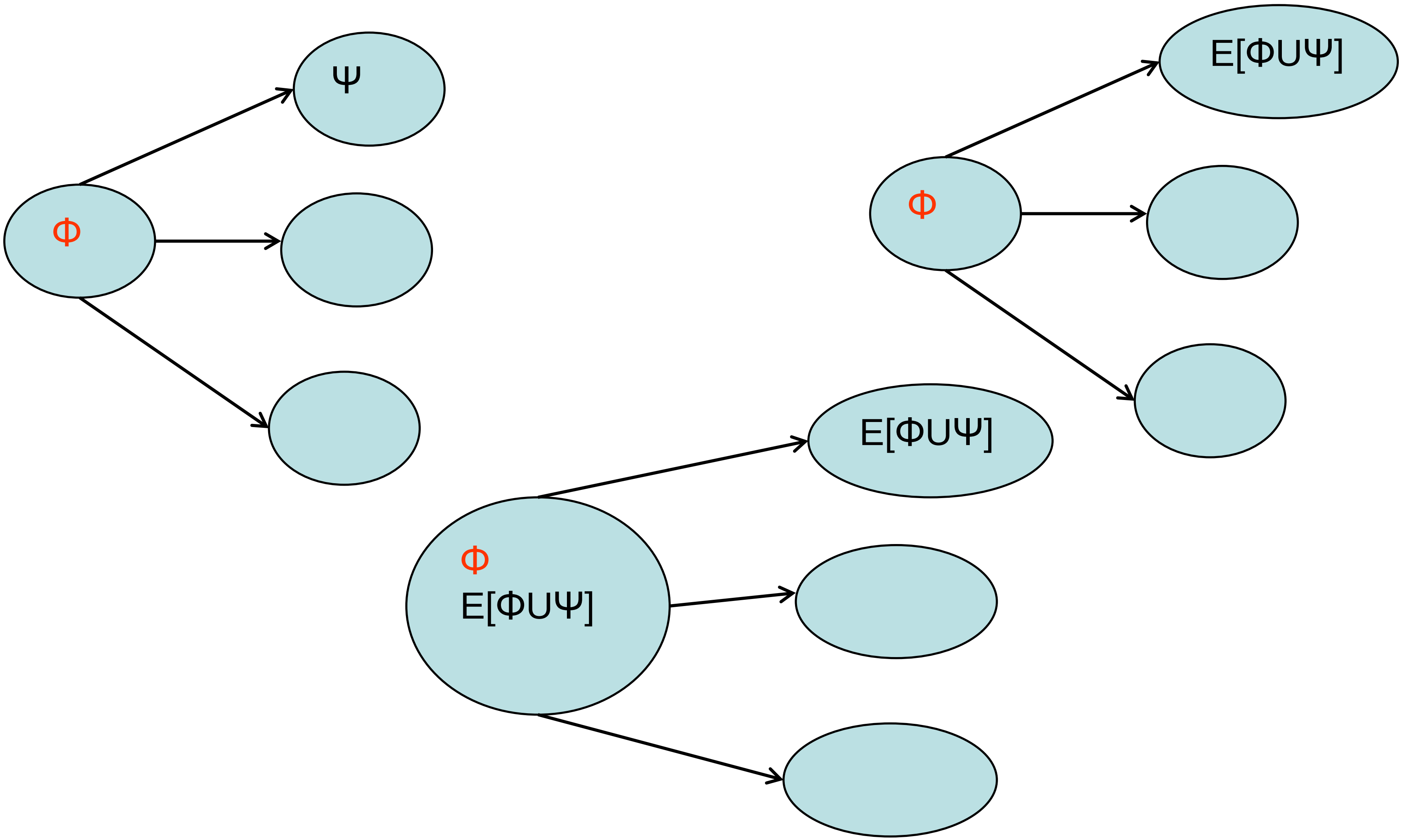


Checking CTL Formula

Algorithm for Checking E(Φ U Ψ)

- assume formulas Φ and Ψ have been verified
- E(Φ U Ψ) is true at a node if there is a path from the node to a Ψ -labeled node, and at every node along that partial path Φ is labeled but Ψ is not
- A node satisfies E(Φ U Ψ) if Ψ is labeled at the node or Φ but not Ψ is labeled at the node and its successor is either labeled Ψ or E(Φ U Ψ)

Checking CTL Formula: E[ΦUΨ]



Checking CTL Formula

- Algorithm for Checking E(Φ U Ψ)

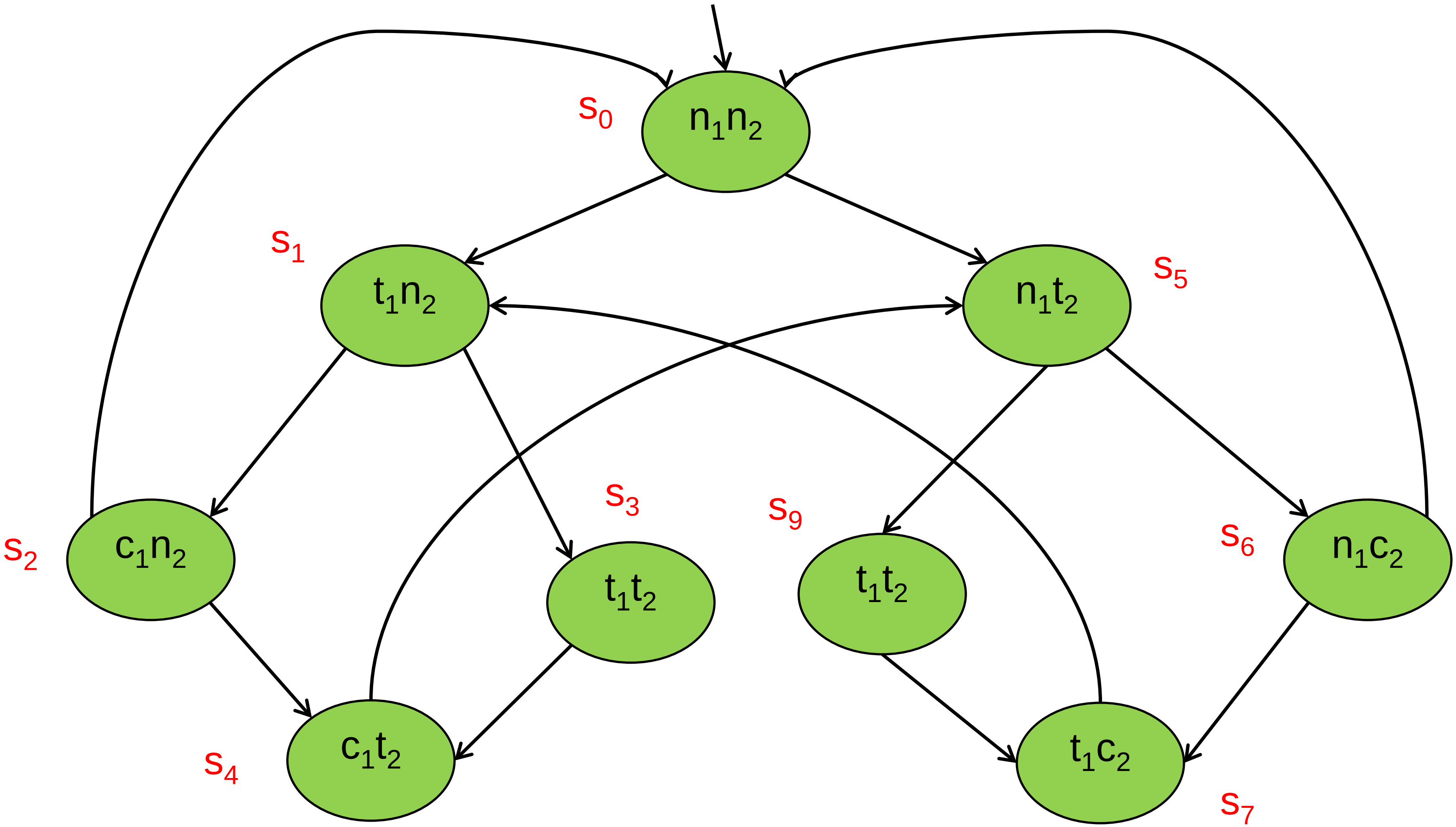
input: a Kripke structure K and a CTL formula E(fUΨ).

output: labeling of the states where E(ΦU Ψ) holds.

Verify_EU(Φ, Ψ): // check CTL formula E(ΦUΨ)

1. M = empty.
2. Add label E(Φ U Ψ) to all states that have label Ψ. Call this set of states L.
3. For every state in L, if there is a predecessor, p, that is not in L and has a label Φ, add label E(Φ U Ψ) to p. Add p to set M. Set M consists of newly added nodes.
4. Set L = M and M = empty.
5. Repeat steps 3 and 4 until L is empty.

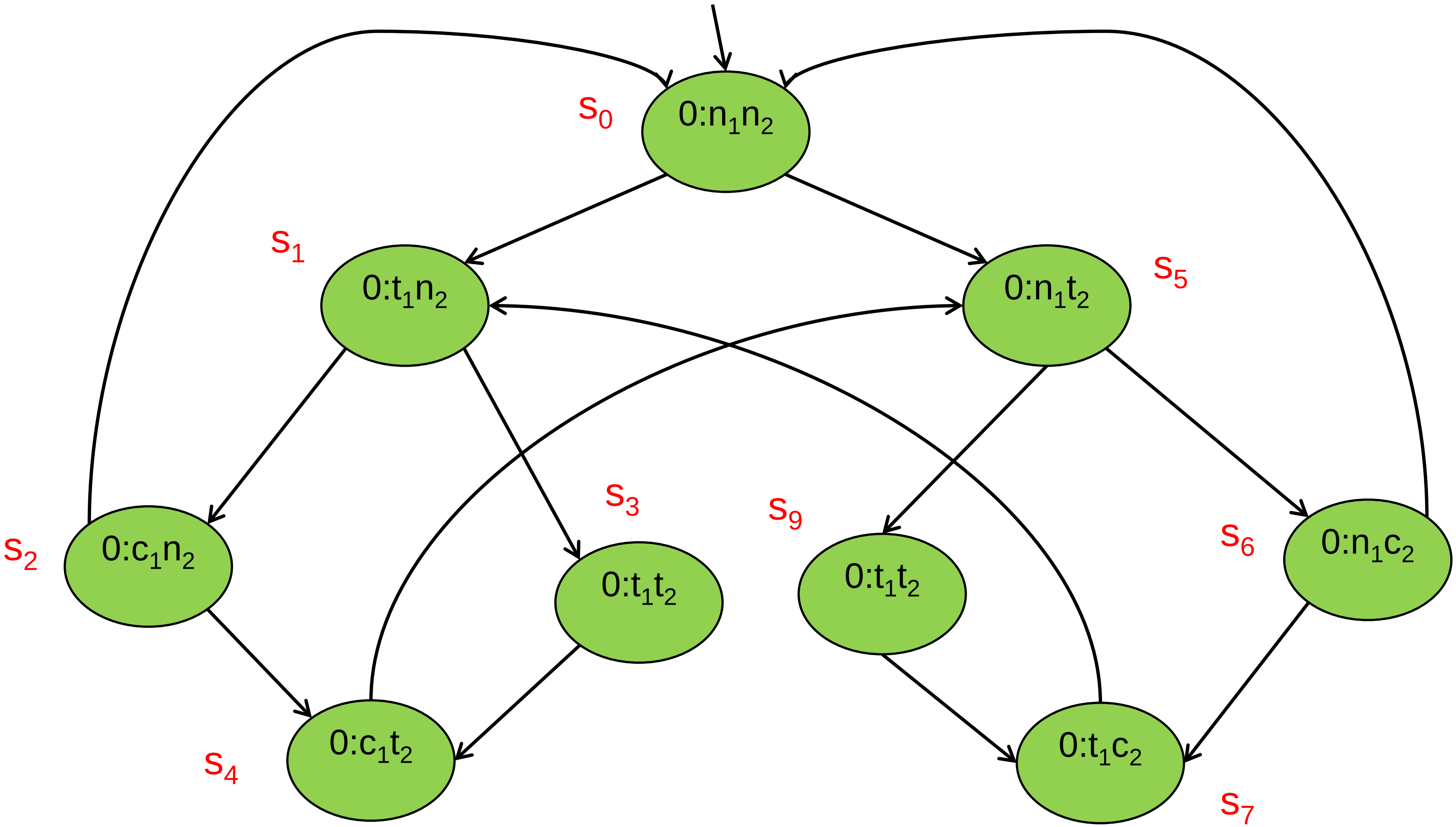
Mutual Exclusion: Implementation 2



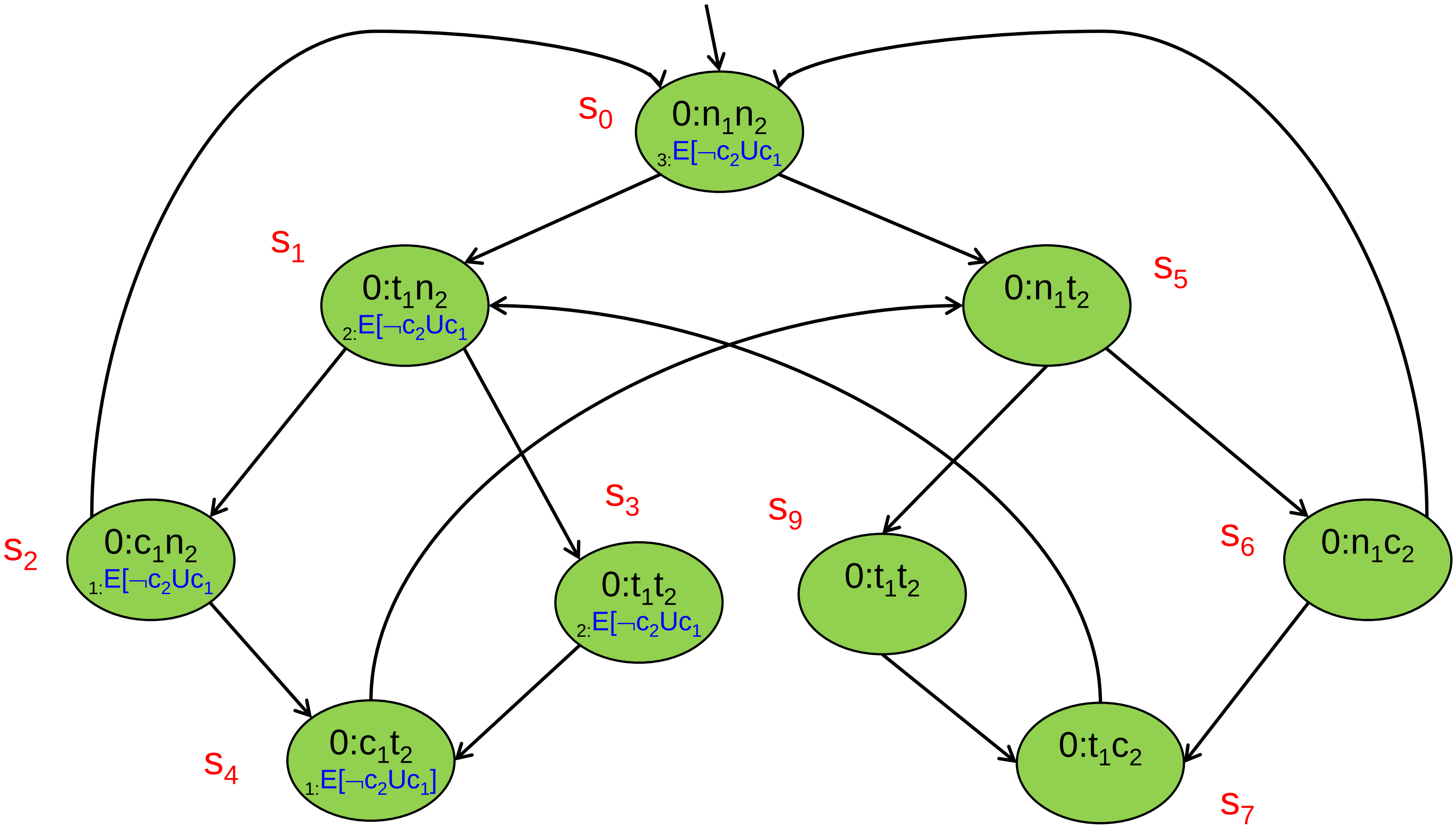
Mutual Exclusion: Properties

1. Safety
   - Only one process in the critical section
   - ➢ $AG\ \neg(c_1 \wedge c_2)$
2. Liveness
   - Whenever any process request to enter its critical section, it will eventually be permitted
   - ➢ $AG\ (t_1 \rightarrow AFc_1)$
3. Non Blocking
   - For every state satisfying n_1, there is a successor satisfying t_1
   - ➢ $AG\ (n_1 \rightarrow EX\ t_1)$
4. No strict sequencing
   - ➢ $EF(c_1 \wedge E[c_1\ U\ (\neg\ c_1 \wedge E\ [\neg\ c_2\ U\ c_1]))])$

Mutual Exclusion: Implementation 2

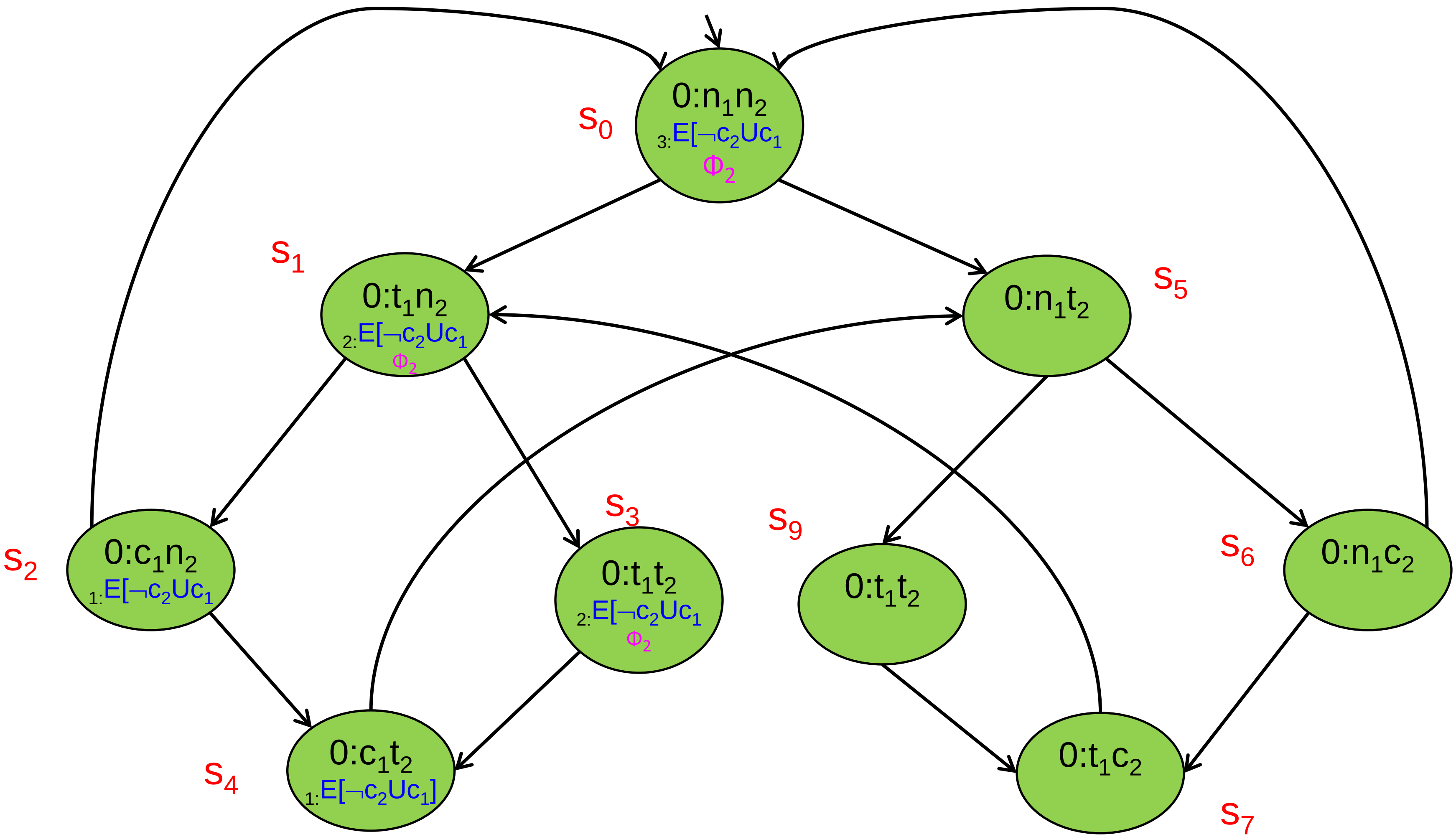


Mutual Exclusion: Property 4



Mutual Exclusion: Property 4

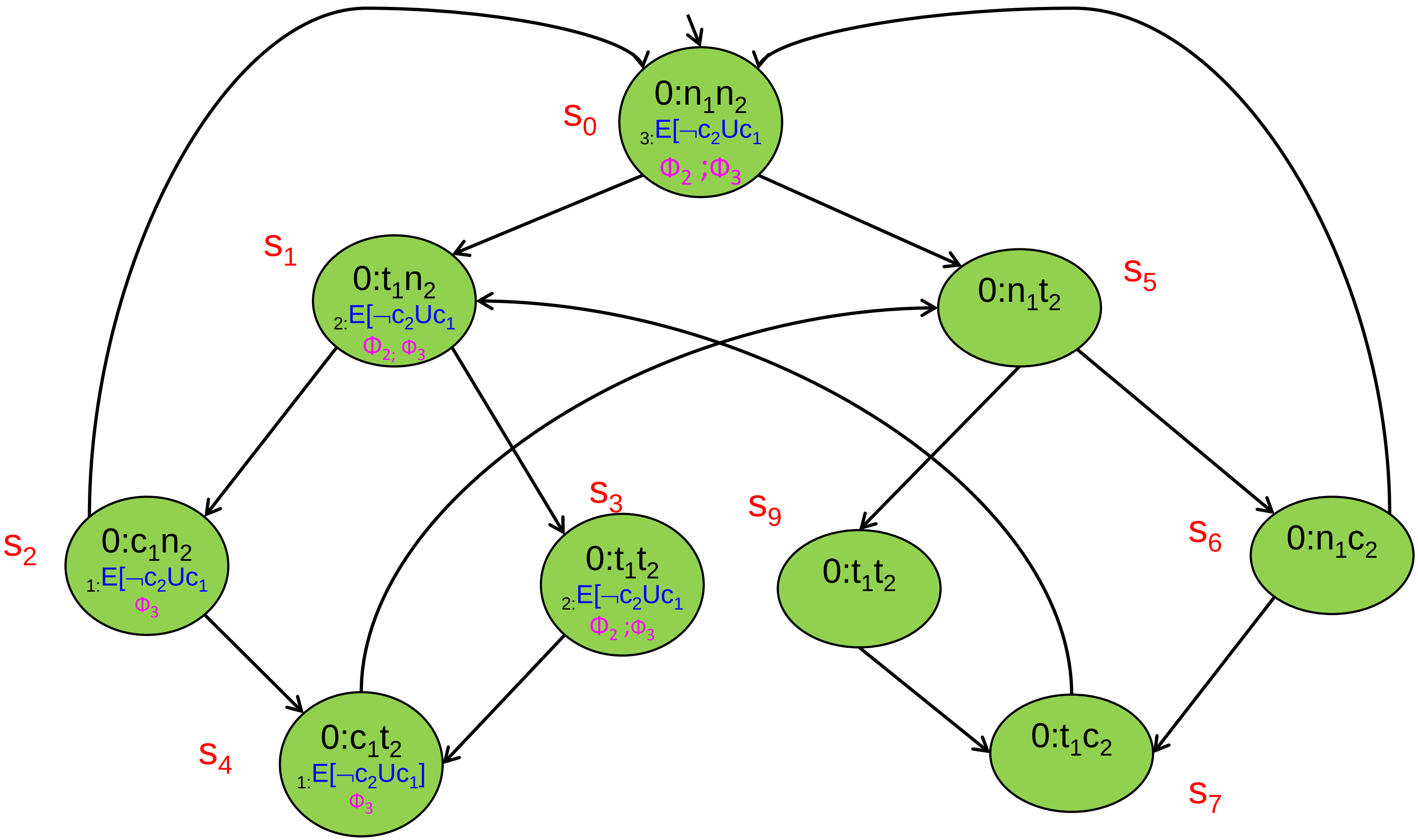
$\Phi_2 = \neg c_1 \wedge E[\neg c_2 \, U \, c_1]))]$



Mutual Exclusion: Property 4

Φ_3=E(c1 U $\neg$ c_1 $\wedge$ E [$\neg$ c_2 U c_1])]

Φ_3= E[c1 U Φ_2]

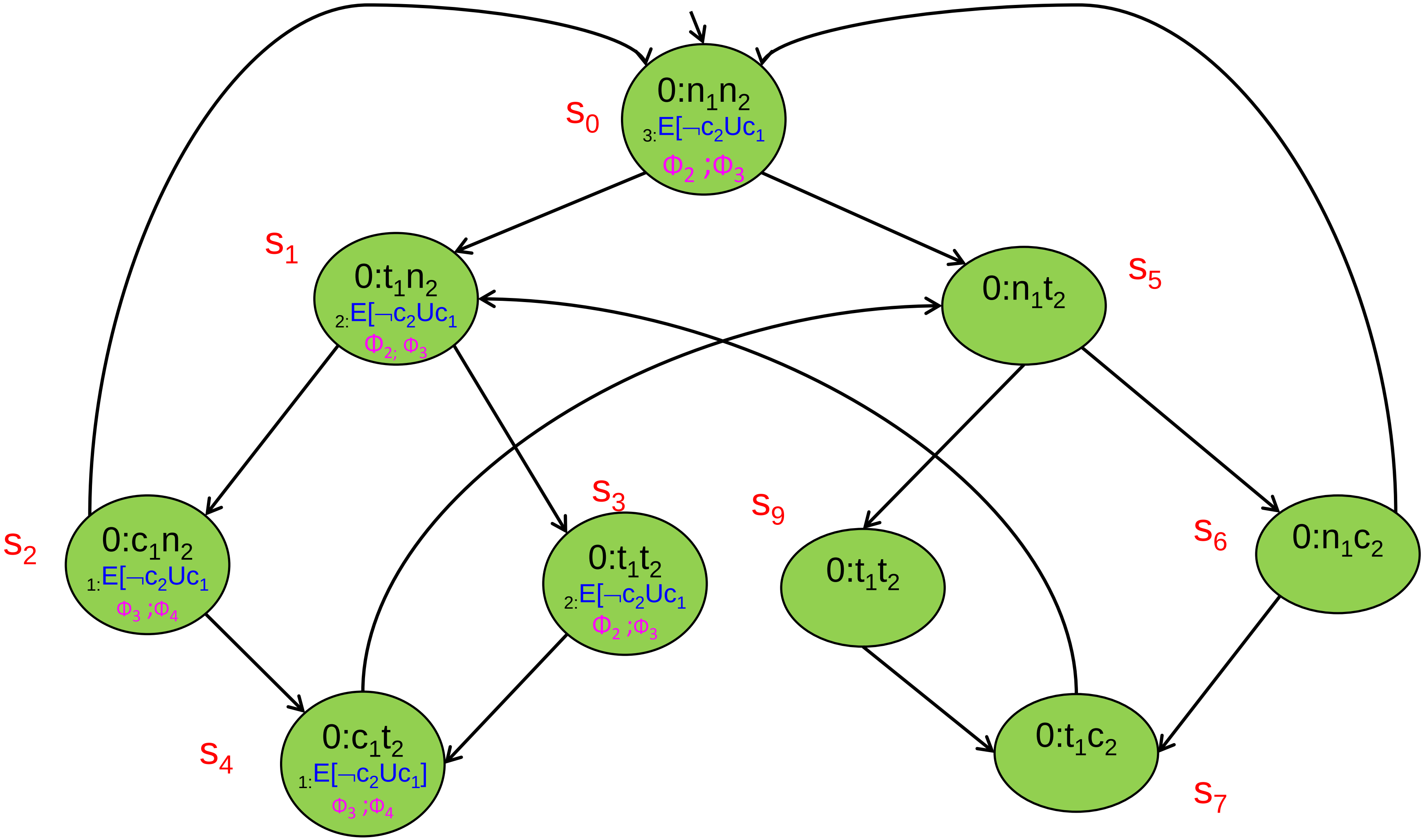


CADSL

Mutual Exclusion: Property 4

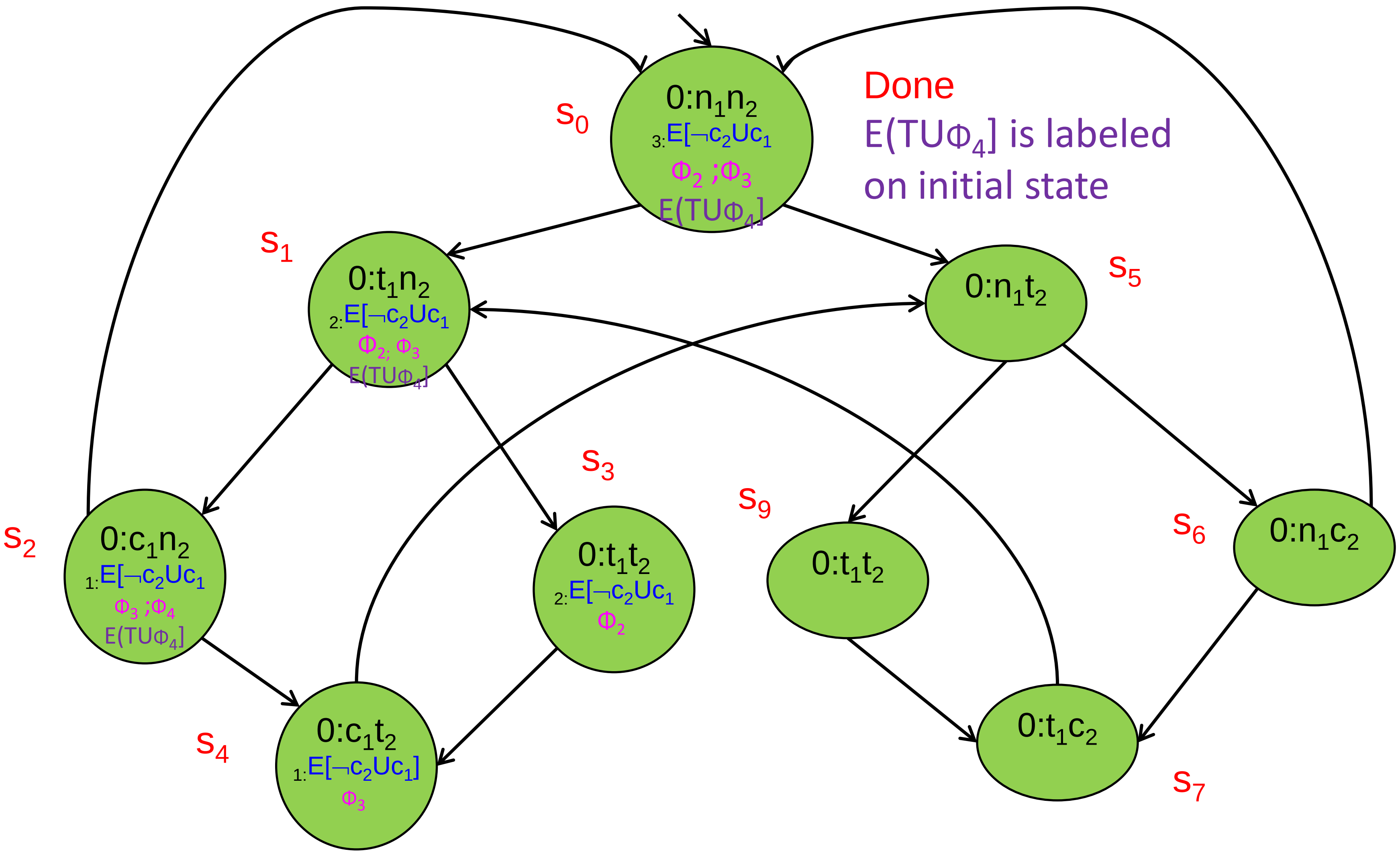
$\Phi_4 = c_1 \wedge E(c1\ U \neg c_1 \wedge E[\neg c_2\ U\ c_1]))$

$\Phi_4 = c_1 \wedge \Phi_3$



Mutual Exclusion: Property 4

$EF\Phi_4 = E(TU\Phi_4]$



Done

$E(TU\Phi_4]$ is labeled on initial state

SAT Formulation

ϕ is T : return S

ϕ is $\neg$T : return null

ϕ is atomic : return $\{s \in S \mid \phi \in L(s)\}$

ϕ is $\neg\phi_1$: return S - SAT(ϕ_1)

ϕ is $\phi_1 \wedge \phi_2$: return $\mathrm{SAT}(\phi_1) \cap SAT(\phi_2)$

ϕ is $\phi_1 \vee \phi_2$: return $\mathrm{SAT}(\phi_1) \cup SAT(\phi_2)$

ϕ is $\phi_1 \rightarrow \phi_2$: return $\mathrm{SAT}(\neg\phi_1 \vee \phi_2)$

SAT Formulation

ϕ is $\mathrm{AX}\phi_1$: return $\mathrm{SAT}(\neg \mathrm{EX} \neg \phi_1)$

ϕ is $\mathrm{EX}\phi_1$: return $\mathrm{SAT}_{EX}(\phi_1)$

ϕ is $\mathrm{A}[\phi_1 \mathrm{U} \phi_2]$: return $\mathrm{SAT}(\neg(\mathrm{E}[\neg \phi_2 U (\phi_1 \wedge \phi_2)] \vee EG(\neg \phi_2))$

ϕ is $\mathrm{E}[\phi_1 \mathrm{U} \phi_2]$: return $\mathrm{SAT}_{\mathrm{EU}}(\phi_1, \phi_2)$

ϕ is $\mathrm{EF}\phi_1$: return $\mathrm{SAT}(\mathrm{E}(\mathrm{T}\ \mathrm{U}\ \phi_1))$

ϕ is $\mathrm{AG}\phi_1$: return $\mathrm{SAT}(\neg \mathrm{AF} \neg \phi_1)$

ϕ is $\mathrm{AF}\phi_1$: return $\mathrm{SAT}_{AF}(\phi_1)$

ϕ is $\mathrm{AG}\phi_1$: return $\mathrm{SAT}(\neg \mathrm{EF} \neg \phi_1)$

Model Checking Algorithms: EX

- Function $SAT_{EX}(\Phi)$
  - /* Determines the set of states satisfying $EX\Phi$ */
- Local var X, Y
- Begin
  - $X := SAT(\Phi)$
  - $Y := pre_{\exists}(X)$
  - Return Y
- End

Model Checking Algorithms: EX

- Function $SAT_{AF}(\Phi)$ /* Determines the set of states satisfying AFΦ */
- local var X, Y
- Begin
  - X := S
  - Y := SAT (Φ)
  - repeat until X = Y
  - begin
    - X := Y
    - Y := Y U $pre_{\forall}$ (Y)
  - end
  - return Y

Model Checking Algorithms: EU

- Function $SAT_{EU}(\phi, \Psi)$ /* Determines the set of states satisfying EU*/
- local var W, X, Y
- Begin
  - W := SAT (ϕ)
  - X := S
  - Y := SAT (Ψ)
  - repeat until X = Y
  - begin
    - X := Y
    - Y := Y U(W ∩ $pre_{\exists}$(Y))
  - end
  - return Y
- End

Symbolic Model Checking

- implicit representation
- graphs and their traversal are converted to Boolean functions and Boolean operations
- all model-checking operations are operations on Boolean functions

Thank You