# CS 5090: Software Fault Tolerance – Model Checking CTL Properties

Ali Ebnenasir
Department of Computer Science
Michigan Technological University



## Acknowledgement


- The contents of this Lecture are adapted from the following paper:

"**Automatic verification of Finite State Concurrent Systems Using Temporal Logic Specifications: A Practical Approach**"

By: E.M. Clarke, E.A. Emerson, A.P. Sistla




## Outline

- Problem statement
- Preliminaries
- Model checking algorithm
- Example: Mutual exclusion

MichiganTech

## Problem Statement

- Check whether a CTL formula is true in a Kripke structure M = (S, R, L), where
  - S is a finite set of states
  - R total binary relation on S
  - L is an assignment of atomic propositions to states



MichiganTech

## Structure vs. Computation Trees

- For a structure M and a state $s_0$ in S, there is an infinite computation tree with root $s_0$ such that $s_i \rightarrow s_j$ is an arc in the tree iff $(s_i , s_j)$ is in R



MichiganTech

## Preliminaries

- CTL formulas: $\neg x$, $x \wedge y$, ***AX***z, ***EX***z, ***A***(x U y), ***E***(xUy)
- Subformula: a subsequence of symbols in a formula that is a formula itself
  - Example: ***A*** $((\neg x)$ ***U*** $(y \vee z))$
    - Subformula 1: ***A*** $((\neg x)$ ***U*** $(y \vee z))$
    - Subformula 2: $(\neg x)$
    - Subformula 3: x
    - Subformula 4: $(y \vee z)$
    - Subformula 5: y
    - Subformula 6: z

MichiganTech

## Preliminaries

- labelled(s, f): a Boolean function that returns
  - True ***iff*** formula f is in L(s)
  - False ***iff*** formula f is not in L(s)
- add_label(s, f): adds f to L(s)
- We need an algorithm that labels each state with the subformulas that are true in that state



MichiganTech

## Algorithm for Checking ***A***(x U y)

- stack: keeps unexplored states
- marked(s): returns true iff s has been visited
- Algorithm:
  - Begin
    - Empty the Stack
    - For all states s in S, **marked**(s) := false;
    - For all states in s do
      - If ¬**marked**(s) then **explore**(f,s,b);
  - End



MichiganTech

## Recursive Labeling Algorithm – Explore(f, s, b)

- Explore(f, s, b):
  - f is a CTL formula
  - s is a state
  - b is a Boolean variable representing the result

## Recursive Labeling Algorithm

- If ***marked***(s) then
  - If ***stacked***(s) then b := false; return;
  - If ***labeled***(s) then b := true; return;
  - b := false; return;
- ***marked***(s) := true;
- If labeled(s, y) then // Checking ***A***(x U y)
  - add_label(s, ***A***(x U y))
  - b := true;
  - Else If (¬labeled(s, x)) then b := false; return;

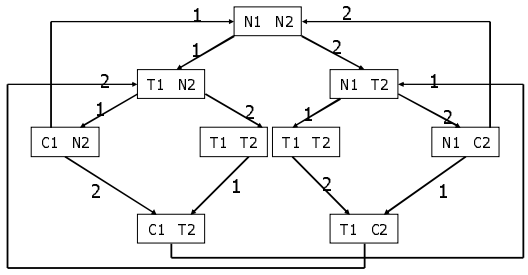


## Recursive Labeling Algorithm – Cont'd

- Push(s);
- For all s' such that s' is a successor of s do
  - Explore(***A***(x U y), s', b');
  - If (¬b') then
    - Pop;
    - b := false; return
- Pop; // get here if ***A***(x U y) holds in all successors of s
- add_label(s, f);
- b := true; return



## Example: Mutual Exclusion State Transition Graph



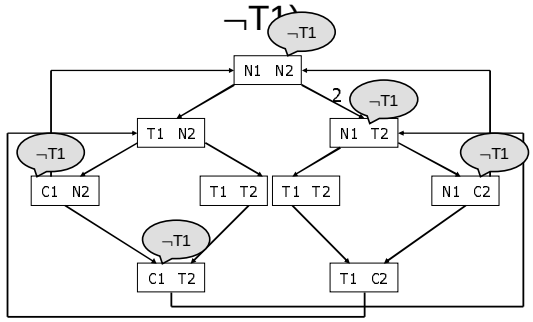

**Check (T1 => AFC1 )**

## Example: Mutual Exclusion

- Subformulas:
  - Subformula 1 → ¬T1 ∨ A(true U C1)
  - Subformula 2 → ¬T1
  - Subformula 3 → T1
  - Subformula 4 → A(true U C1)
  - Subformula 5 → true
  - Subformula 6 → C1



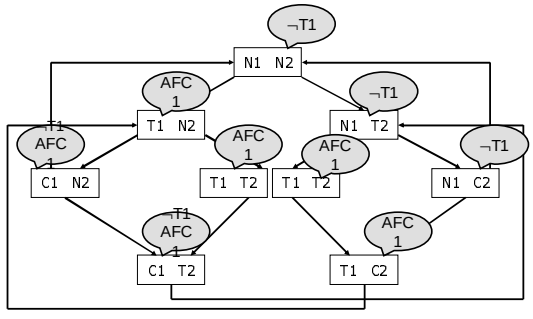
## Example: Exploring T1 (and ¬T1)




**Check (T1 => AFC1 )**

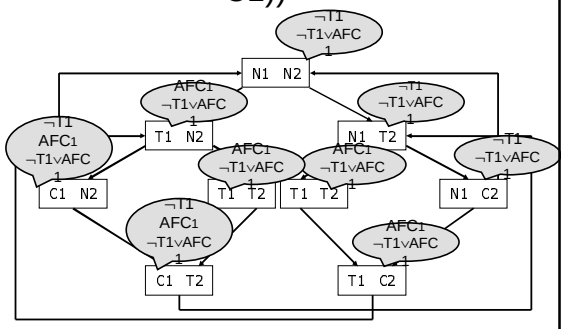
14

## Example: Exploring C1 (i.e., ***A***(true U C1))




15

## Example: Exploring C1 (i.e., ***A***(true U C1))




16

**<N1, N2> |= AG(T1 => AFC1 )**

## Theorem

- There exists an algorithm for determining whether or not a CTL formula is true in a Kripke structure M=(S, R, L).
- Time complexity:
  O(sizeof(formula) . sizeof(S) . sizeof(R))
  Polynomial in the set of states

S/W Fault-Tolerance – Ebnenasir – Spring 2008