Lecture # 15

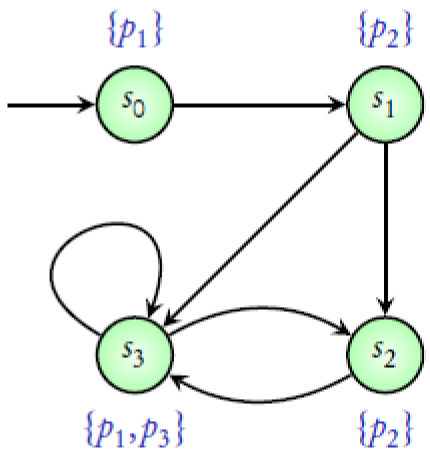
Computation Tree Logic

Computation Tree

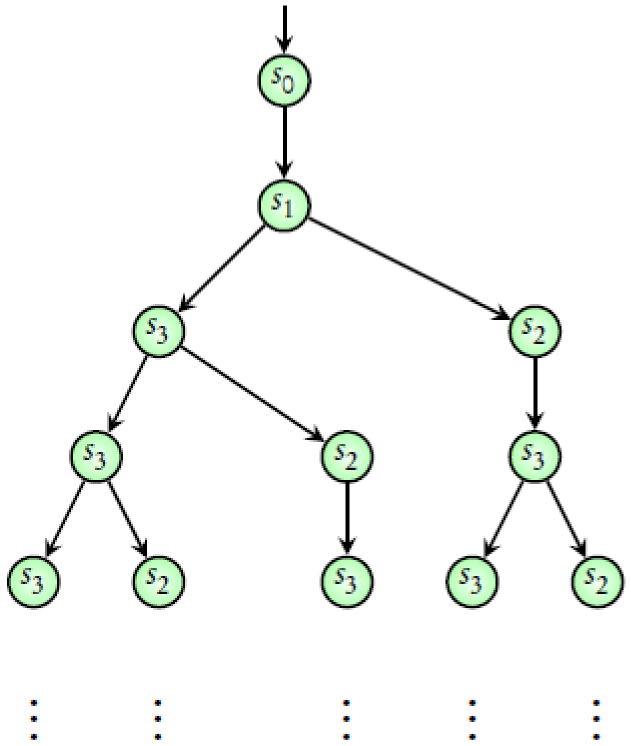
A tree view of the transition system..

..obtained by repeatedly unfolding it

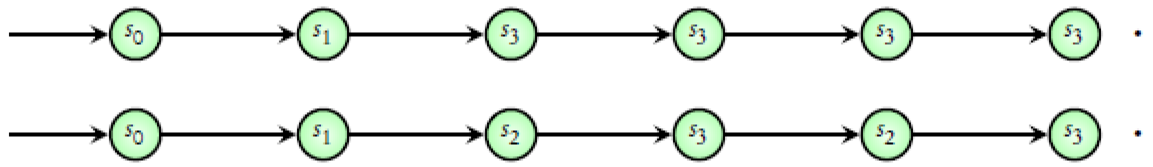
Transition System



Computation tree



Paths



s_0 s_1 s_3 s_3 s_3 s_3
s_0 s_1 s_2 s_3 s_2 s_3

Traces

$\{p_1\}\{p_2\}\{p_1,p_3\}\{p_1,p_3\}\{p_1,p_3\}\{p_1,p_3\} \ldots$

$\{p_1\}\{p_2\}\{p_2\}\{p_1,p_3\}\{p_2\}\{p_1,p_3\}\{p_2\}\{p_1,p_3\} \ldots$

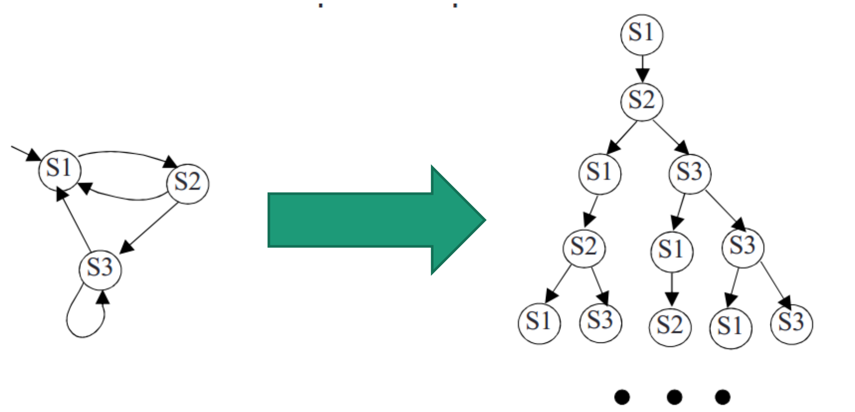
Computation Tree Logic (CTL)

CTL Syntax: each temporal operator is now prefixed by one of the following path operators

A – ‘on all future paths starting here’
E – ‘on some future path starting here’

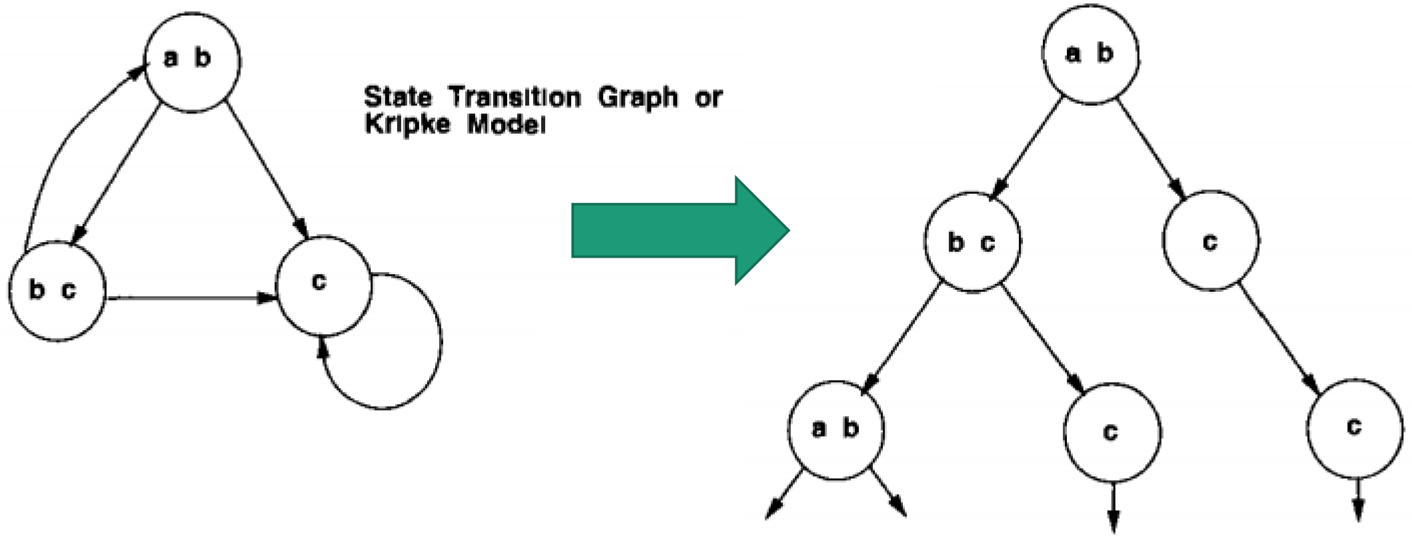
Computation Tree Logic (CTL)

Conceptually, CTL* formulas describe properties of *computation trees* The tree is formed by designating a state in a Kripke structure as the *initial state* and then unwinding the structure into an infinite tree with the designated state at the root, as illustrated in Figure 3 1 The computation tree shows all of the possible executions starting from the initial state.



Kripke Structure

Tree of computation



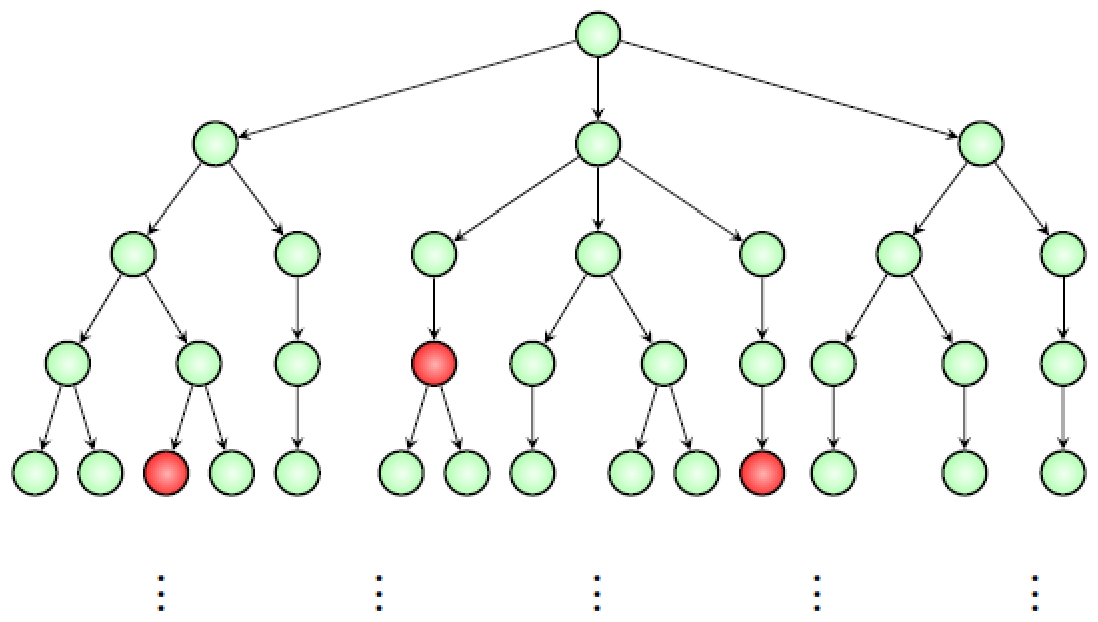
Unwind State Graph to obtain Infinite Tree

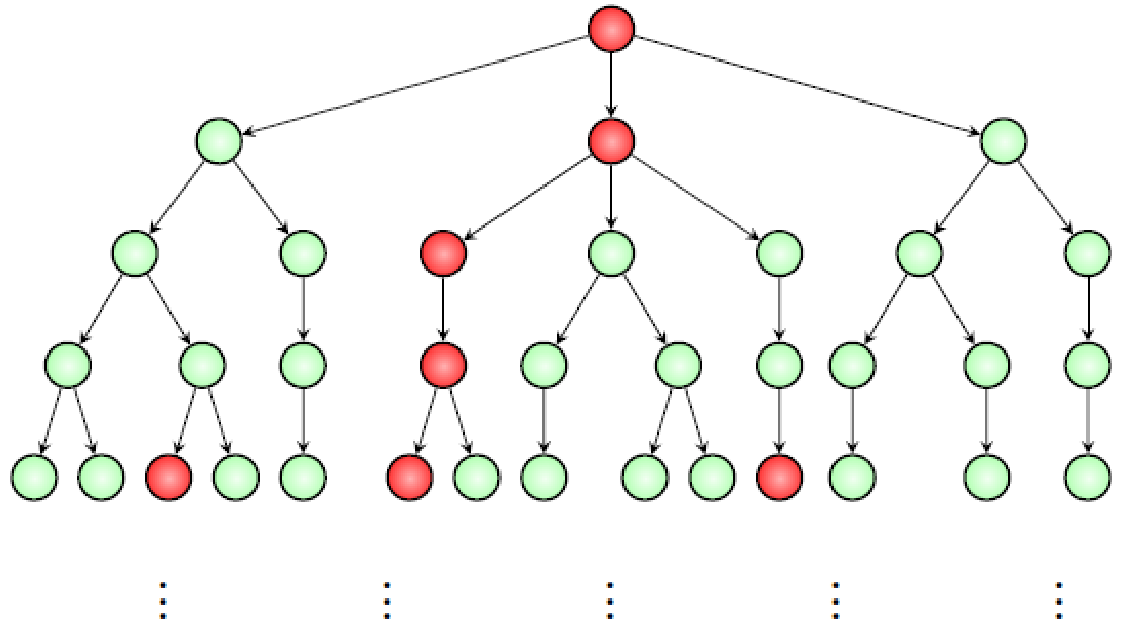
Computation Tree Logic (CTL)

In CTL* formulas are composed of *path quantifiers* and *temporal operators*. The path quantifiers are used to describe the branching structure in the computation tree. There are two such quantifiers **A** ("for all computation paths") and **E** ("for some computation path"). These quantifiers are used in a particular state to specify that all of the paths or some of the paths starting at that state have some property.

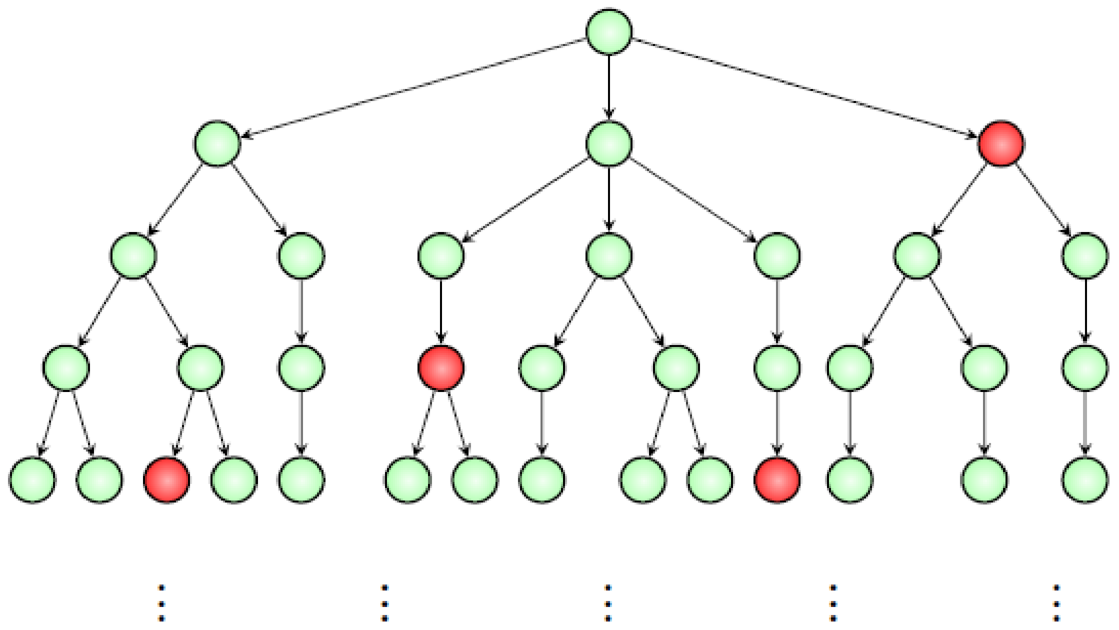
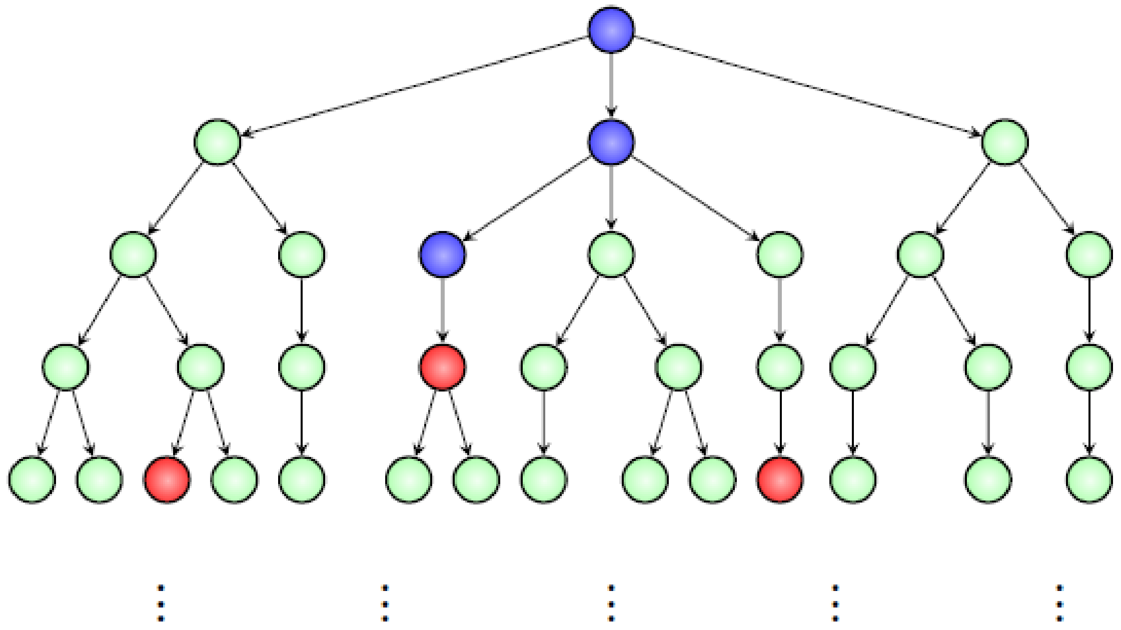
- **X** ("next time") requires that a property holds in the second state of the path.
- The **F** ("eventually" or "in the future") operator is used to assert that a property will hold at some state on the path.
- **G** ("always" or "globally") specifies that a property holds at every state on the path.
- The **U** ("until") operator is a bit more complicated since it is used to combine two properties. It holds if there is a state on the path where the second property holds, and at every preceding state on the path, the first property holds

Computation Tree Logic (CTL)

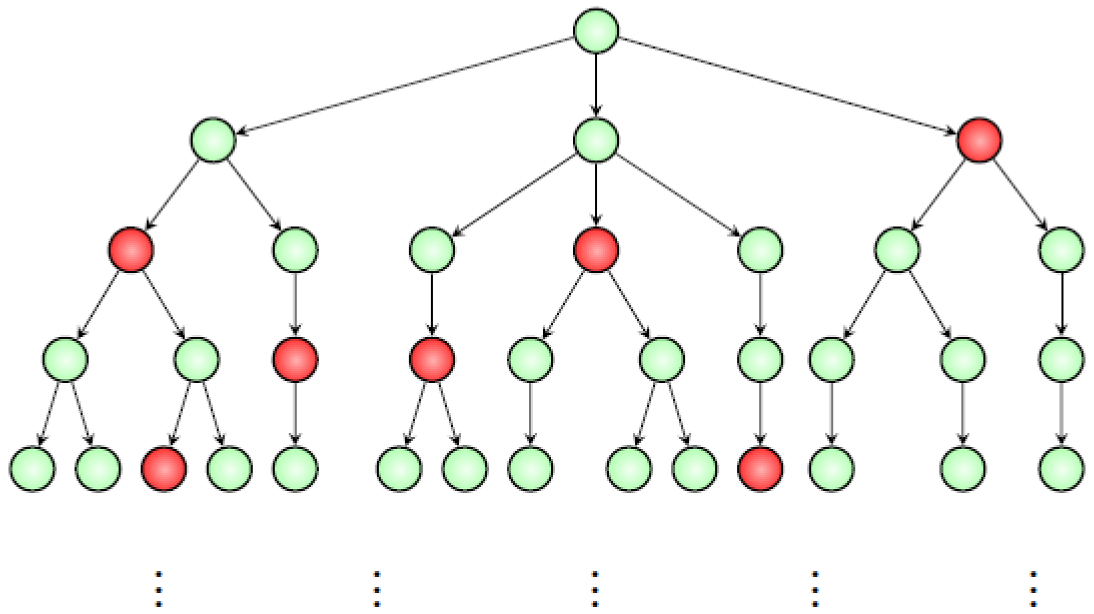


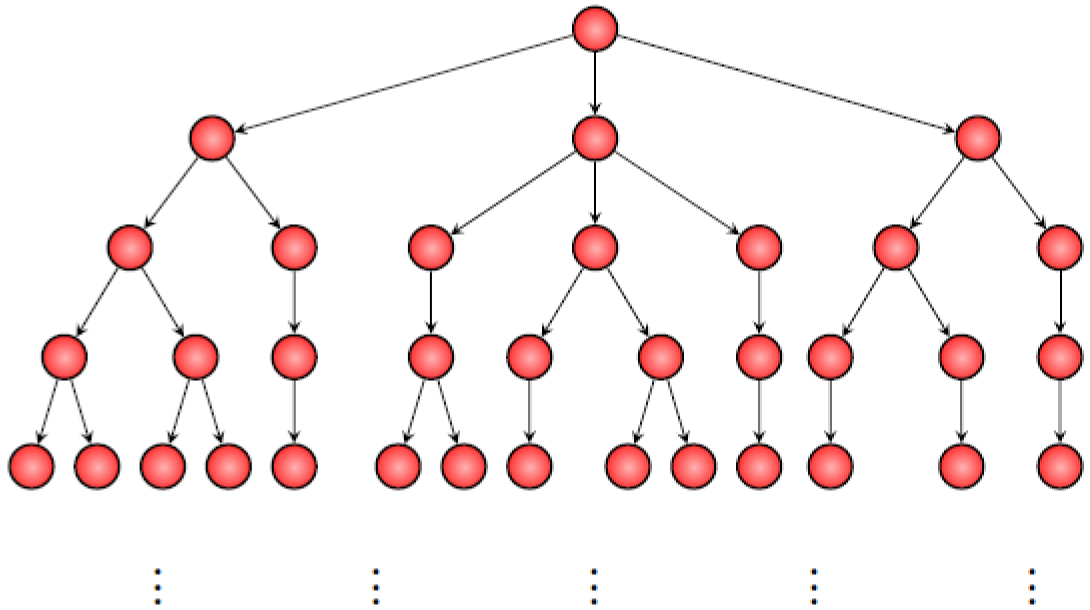


Computation Tree Logic (CTL)

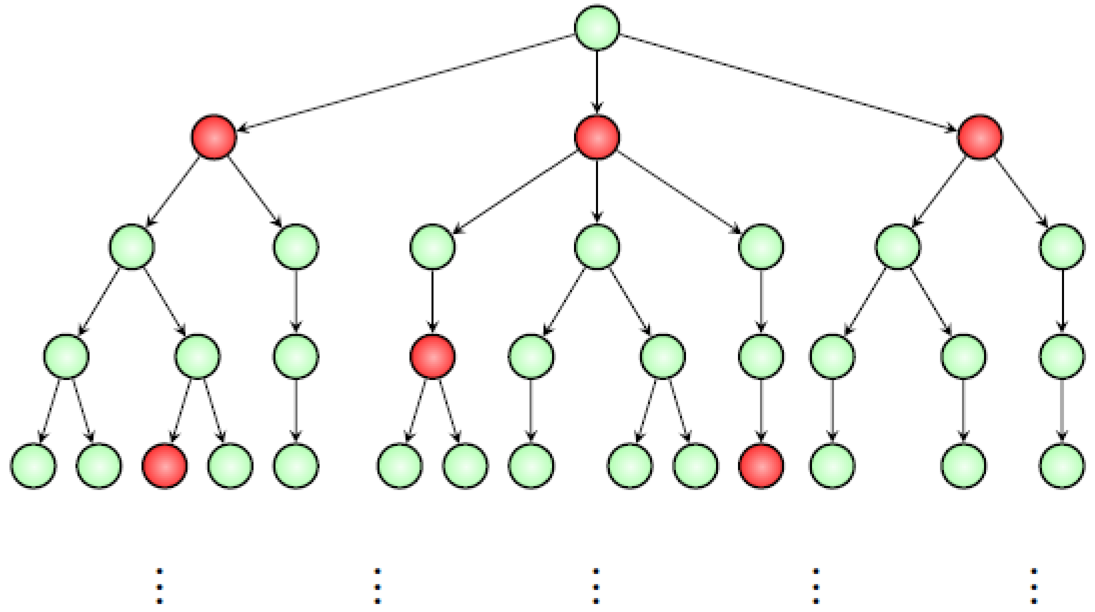



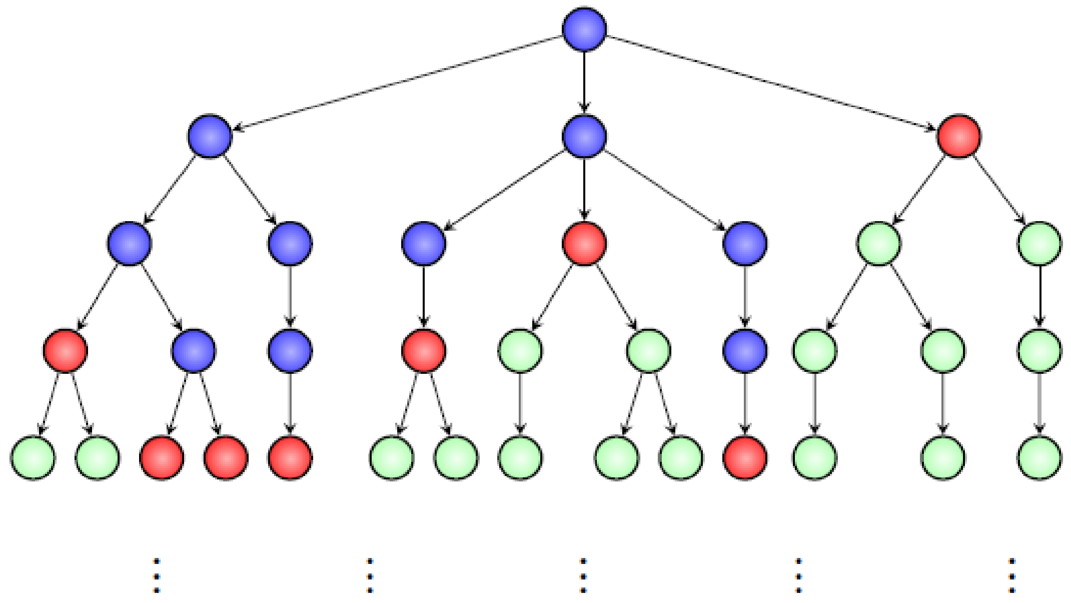
Computation Tree Logic (CTL)





Computation Tree Logic (CTL)





Computation Tree Logic (CTL)

Properties of trees

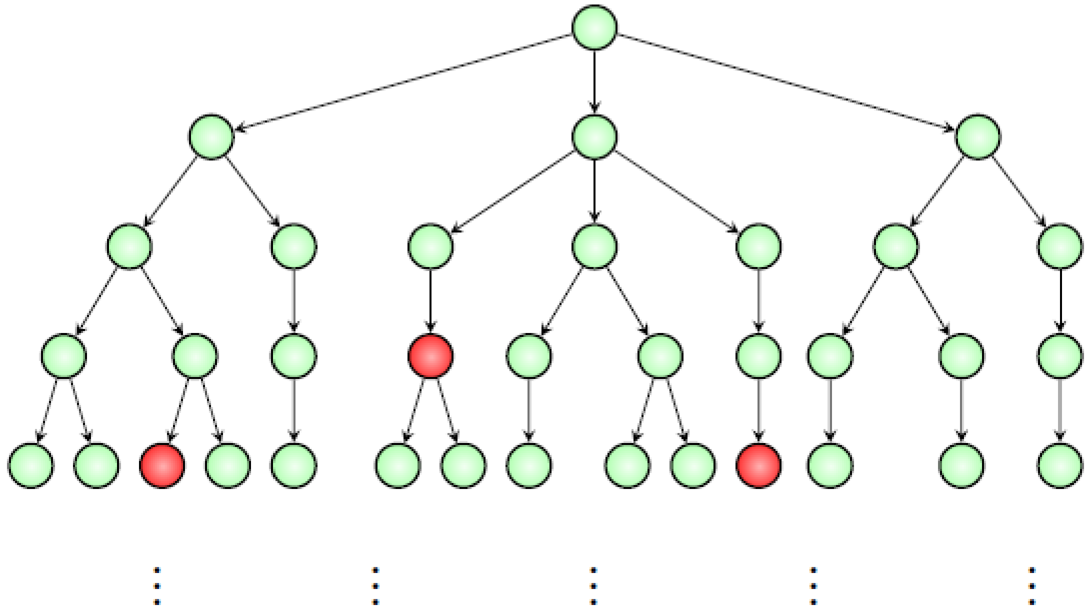
- Exists a path satisfying path property ϕ : $\mathbf{E}\ \phi$
- All paths satisfy path property ϕ : $\mathbf{A}\ \phi$

Coming next: Mixing **A** and **E**

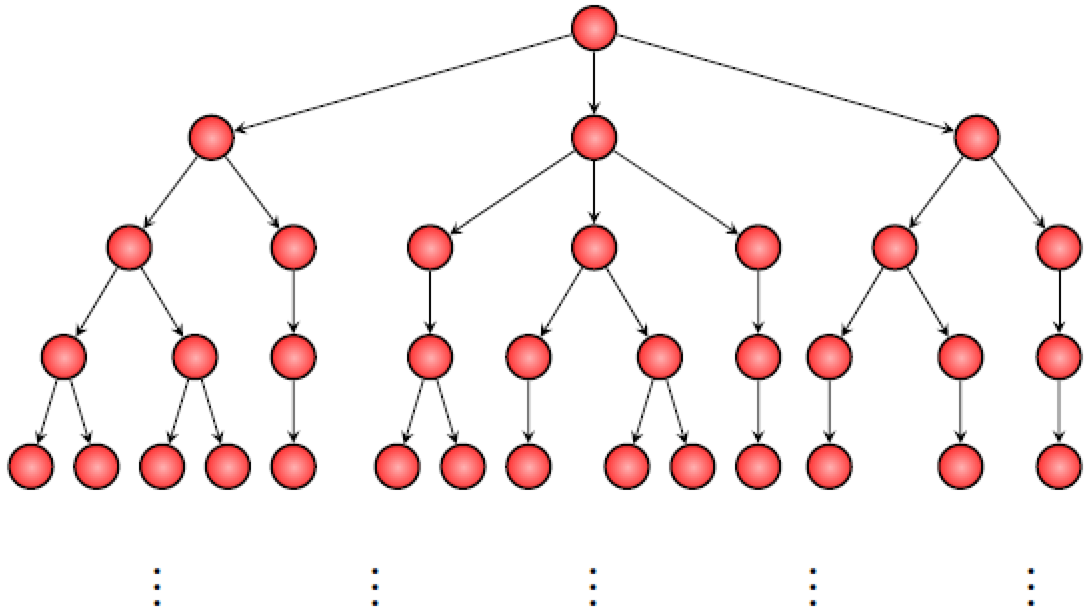
Computation Tree Logic (CTL)

Recap..

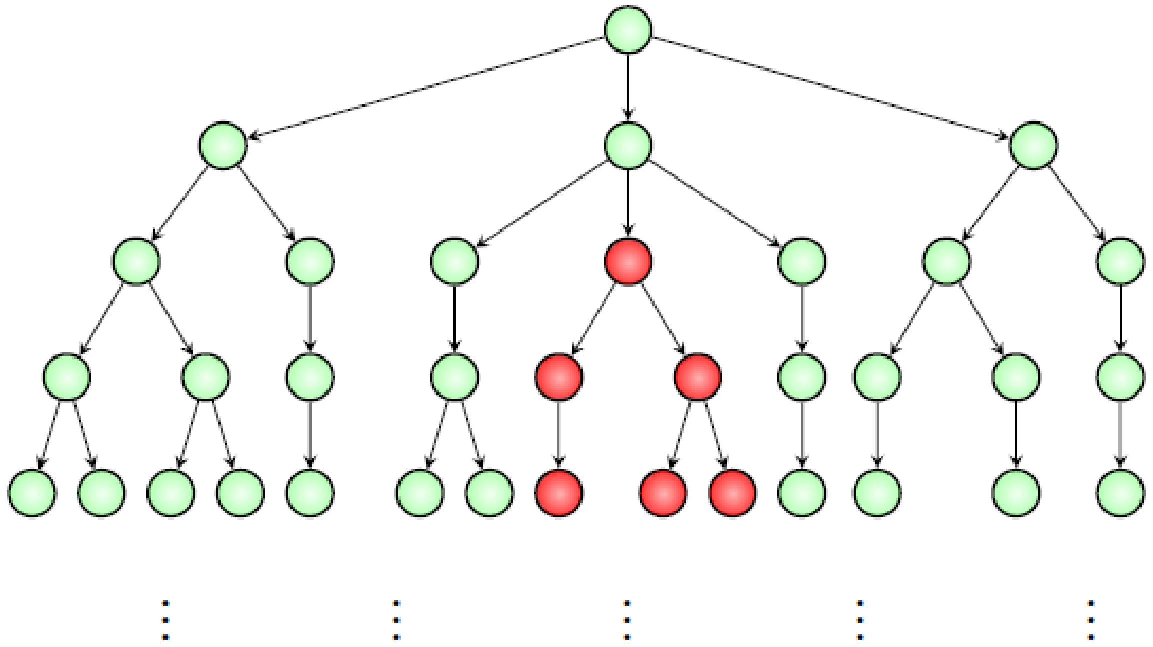
Exists a path satisfying F(*red*) : E F (*red*)



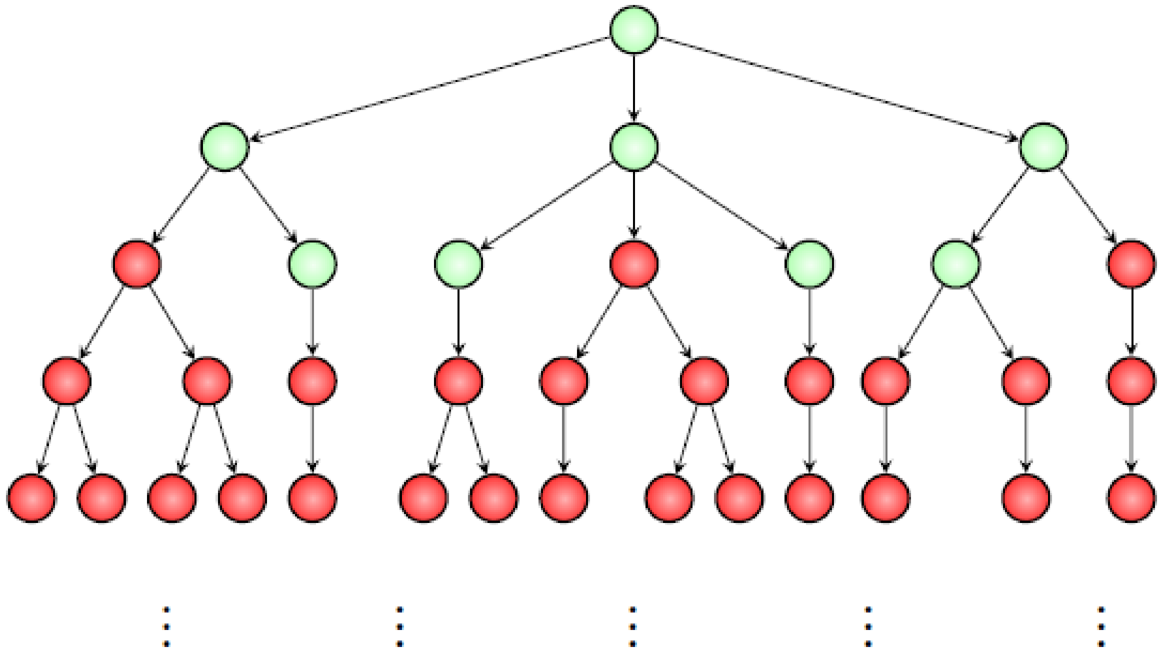
All paths satisfy G(*red*) : A G(*red*)



Computation Tree Logic (CTL)



There exist a path such that FAG (red) is true
There exist a path which reaches a state where AG (red) is true

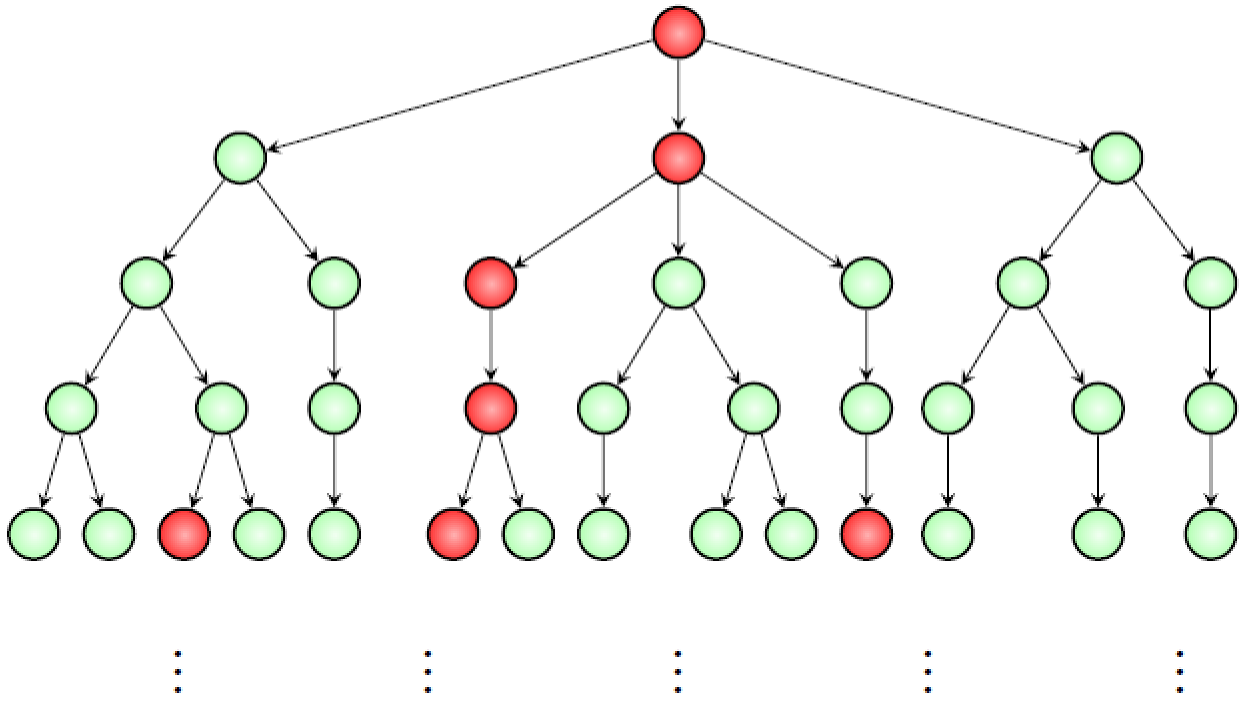


In all paths, there exists a state where A G red is true
In all paths, there exists a state from which all paths satisfy G red
In all paths, there exists a state such that every state in the subtree below it contains red

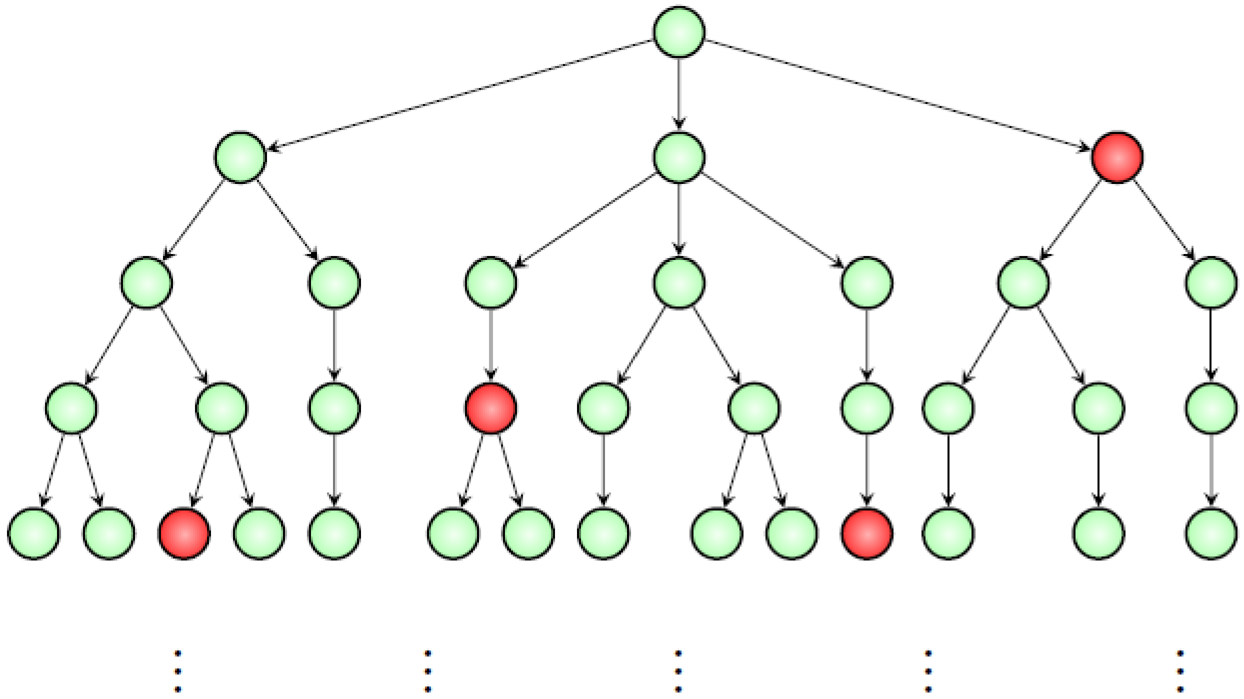
Computation Tree Logic (CTL)

Recap..

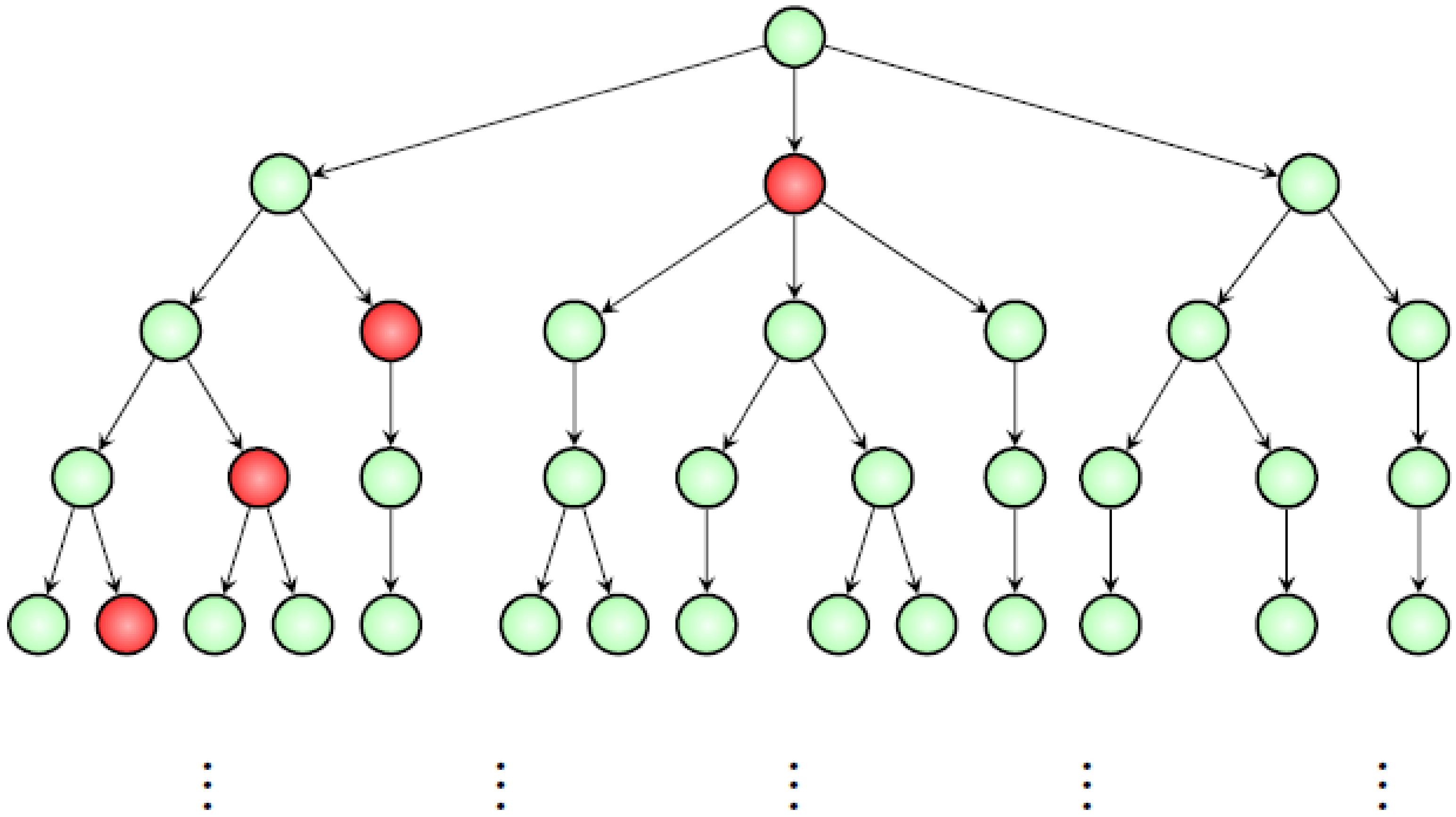
Exists a path satisfying G(*red*) : E G (*red*)



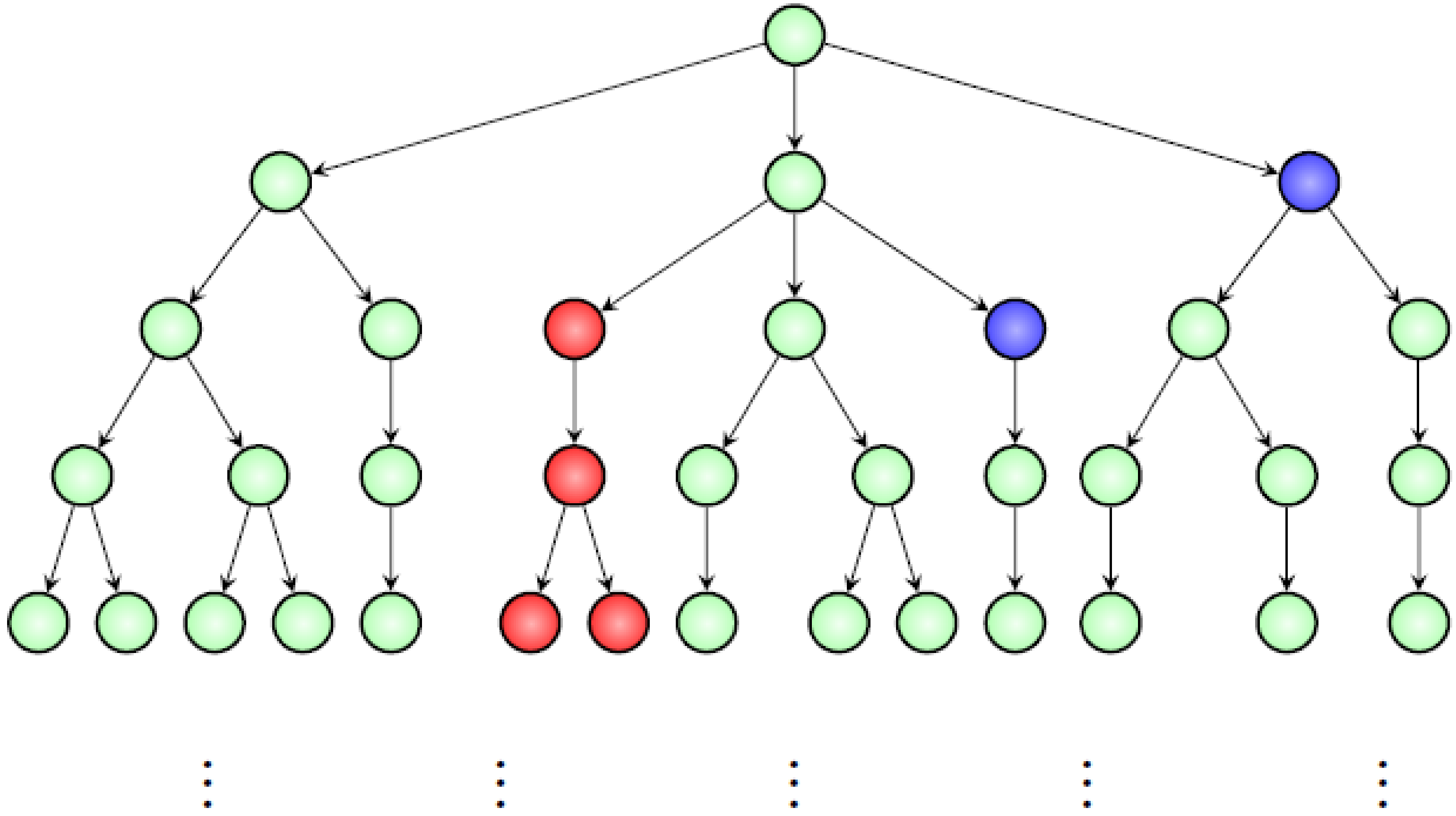
Exists a path satisfying X(*red*) : E X (*red*)



Computation Tree Logic (CTL)



Computation Tree Logic (CTL)



Temporal Logic Gets Everywhere

- specification and verification of (dynamic) programs
- specification and verification of distributed and
- concurrent programs
- representation of tense in natural language
- characterizing temporal database queries
- verification of finite state models derived from ‘real’
- systems (model checking)
- agent theory
- direct execution
- real-time analysis
- temporal data mining
- exploring the limits of decidability
- …….

REFERENCES :

1. Michael Huth, Mark Ryan-Cambridge, Logic in computer science_ modelling and reasoning about systems University Press (2004) - Chapter 5