Computation Tree Logic

Readings: Huth, Ryan, 3.2-3.4

Computational tree logic - propositional branching time logic, permitting explicit quantification over all possible futures.

Fixed set of atomic formulas (p, q, r), standing for atomic descriptions of a system:
The printer is busy
There are currently no requested jobs for the printer
Conveyer belt is stopped

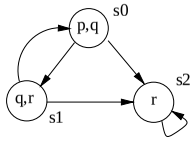
Choice of atomic propositions depends on our intension (but usually does not involve time)

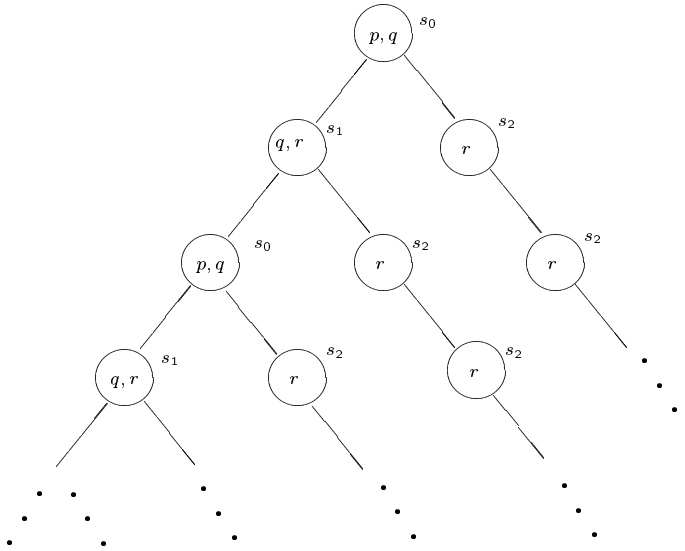
Computation Tree Logic (Cont'd)

Syntax:
1. $\bot$, $\top$, and every atomic proposition is a CTL formula
2. If f and g are CTL formulae, then so are $\neg f$, $f \wedge g$, $f \vee g$, AXf, EXf, A[fUg], E[fUg], AFf, EFf, AGf, EGf.

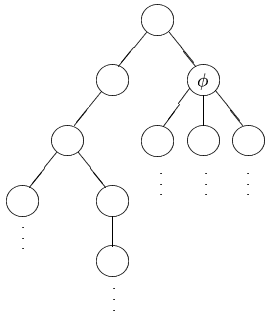
Temporal operators - quantifier (A or E) followed by F (future), G (global), U (until), or X (next).

Where did the "tree" come from?

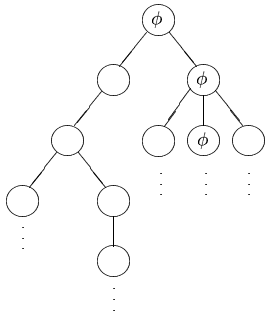




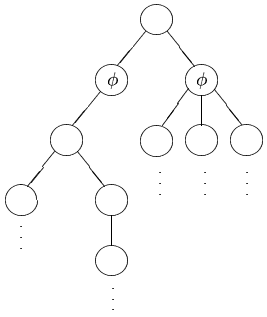
Examples (Part 1)



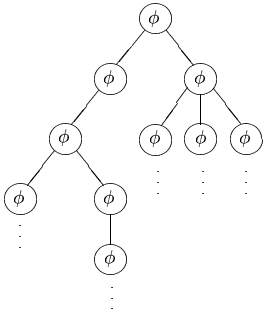
EX (exists next)
"on some path in the next state"



EG (exists global)
"on some path in all states"

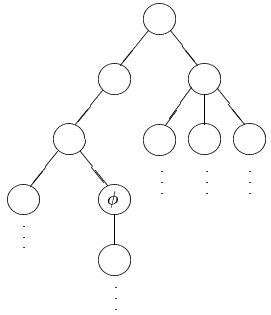


AX (all next)
"on all paths in the next state"

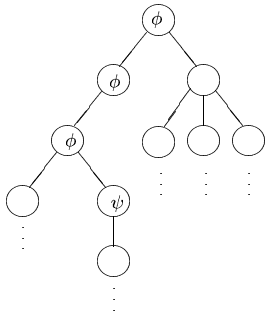


AG (all global)
"on all paths in all states"

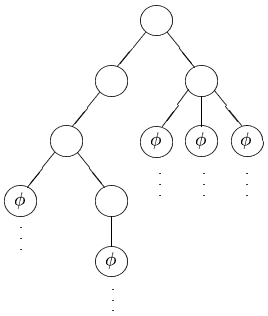
Examples (Part 2)



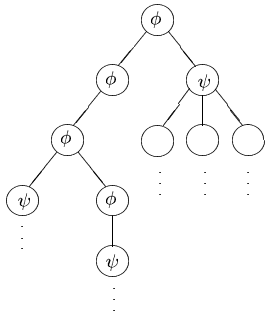
EF (exists future)
"on some path in some state"



EU (exists until)
"on some path until some state"



AF (all future)
"on all paths in some state"



AU (all until)
"on all paths until some state"

CTL Syntax - Cont'd

Which of these are well-formed and which are not:

EG r
FG r
AG AF r
E [A [p_1 U p_2] U p_3]
AF [(r U q) $\wedge$ (p U r)]
EF E[r U q] AF[(r U q) $\wedge$ (p U r)]

Formulas, Subformulas and Parse Trees

Draw parse tree for E[A[p U q] U r]

Draw parse tree for AG($p \rightarrow$ A[p U ($\neg p \wedge$ A[$\neg p$ U q])])

Definition: A subformula of a CTL formula ϕ is any formula ψ whose parse tree is a subtree of ϕ's parse tree.

Semantics of CTL

$M, s \models f$ – means that formula f is true in state s. M is often omitted since we always talk about the same model.

E.g. $s \models x = 1 \wedge \exists n : nat \cdot y = 2 \times n$
means that in state s, variable x has value 1 and variable y has an even natural value.

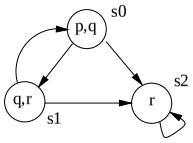
$p = p_0\ p_1\ p_2\ \ldots$ is a path
p_0 is the current state (root)
p_{i+1} is one of p_i's successor states. Then,

AX $f = \forall p \cdot p_1 \models f$
EX $f = \exists p \cdot p_1 \models f$
AG $f = \forall p \cdot \forall i \cdot p_i \models f$
EG $f = \exists p \cdot \forall i \cdot p_i \models f$
AF $f = \forall p \cdot \exists i \cdot p_i \models f$
EF $f = \exists p \cdot \exists i \cdot p_i \models f$
A[f U g] $= \forall p \cdot \exists i \cdot p_i \models g \wedge \forall j \cdot 0 \leq j < i \rightarrow p_j \models f$
E[f U g] $= \exists p \cdot \exists i \cdot p_i \models g \wedge \forall i \cdot 0 \leq j < i \rightarrow p_j \models f$

Note: the i in $\exists i \cdot$ could be 0.

Why Do These Properties Hold?

System:



$M, s_0 \models p \wedge q$
$M, s_0 \models \neg r$
$M, s_0 \models \top$
$M, s_0 \models$ EX$(q \wedge r)$
$M, s_0 \models \neg$ AX$(q \wedge r)$
$M, s_0 \models \neg$ EF $(p \wedge r)$
$M, s_2 \models$ EG r
$M, s_2 \models$ AG r
$M, s_0 \models$ E$[(p \wedge q)$ U $r]$
$M, s_0 \models$ A$[p$ U $r]$

Examples

It is possible to get to a state where *started* holds, but *ready* does not hold. EF($started \wedge \neg ready$)

When a request (of some resource) occurs, it will eventually be acknowledged.

AG ($request \rightarrow$ AF $acknowledge$)

A process is enabled infinitely often on every computation path. AG AF *enabled*

A process will eventually be permanently deadlocked.

AF AG *deadlock*

It is always possible to get to a *restart* state. AG EF *restart*

An elevator does not change direction when it has passengers wishing to go in the same direction.

$\forall f, b \cdot$ AG((floor $= f \wedge$ direction $=$ up $\wedge$ pressed(b) $\wedge\ b > f) \rightarrow$ A[direction $=$ up U floor $= b$])

An elevator can remain idle on the third floor with its doors closed:

AG((floor $= 3 \wedge$ *idle* $\wedge$ door $=$ closed) $\rightarrow$ EG (floor $= 3 \wedge$ *idle* $\wedge$ door $=$ closed))

More Examples

Which situation does this signify:
AG($p \rightarrow$ AF($s \wedge$ AX(AF(t)))

Specify in CTL:
Temperature is always above normal, below normal or normal.

We can always reach a resolution.

Whenever p is followed by q (after some finite amount of steps), then the system enters an "interval" in which no r occurs until t.

Laws

$\neg$ AF f = EG $\neg f$
$\neg$ EF f = AG $\neg f$
$\neg$ AX f = EX $\neg f$
AF f = A[$\top$ U f]
EF f = E[$\top$ U f]
A[$\bot$ U f] = E[$\bot$ U f] = f
AG f = $f \wedge$ AX AG f
EG f = $f \wedge$ EX EG f
AF f = $f \vee$ AX AF f
EF f = $f \vee$ EX EF f
A[f U g] = $g \vee (f \wedge$ AX A[f U g])
E[f U g] = $g \vee (f \wedge$ EX E[f U g])

What Do We Do with CTL?

Mutual Exclusion Problem. Aimed to ensure that two processes do not have access to some shared resource (database, file on disk, etc.) at the same time. Identify *critical* sections and ensure that at most one process is in that section.

Interested in the following properties:
(Safety): The protocol allows only one process to be in its critical section at any time.

Formalization:

Why is this not enough?

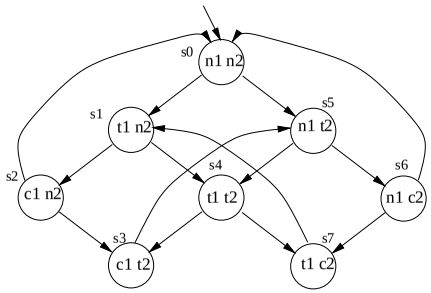
(Liveness): Whenever any process wants to enter its critical section, it will eventually be permitted to do so.

Formalization:

(Non-blocking): A process can always request to enter its critical section.

Formalization:

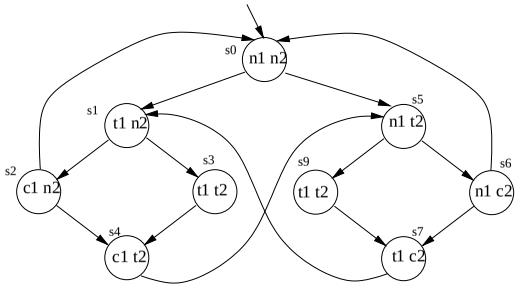
The first modeling attempt



Does it work?

Second modeling attempt

Works!



The problem is a bit simplified (cannot stay in *critical* forever!)

Adequate Sets

Definition: A set of connectives is *adequate* if for every formula there is an equivalent formula with only connectives from that set.

E.G. $\{\neg, \vee\}$ is adequate for propositional logic

Theorem: The set of operators $\bot$, $\neg$ and $\wedge$ together with AF EU and EX are adequate for CTL.

Other adequate sets: {AU, EU, EX}, {AG, AU, AX}.

Kripke Structures (Our Model)

Formula is defined with respect to a model $M = \langle V, S, s_0, R, I \rangle$, where

V is a set of atomic propositions

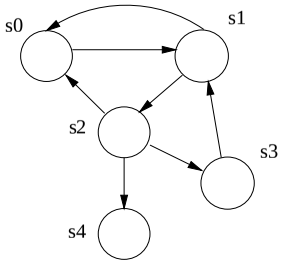
S is a set of states

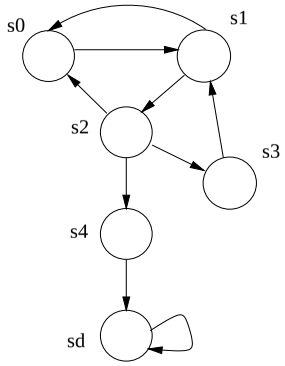
$s_0 \in S$ is the start state

R is a transition relation (every state has a successor)

I is a set of interpretations specifying which propositions are true in each state.

Example:





How to deal with deadlock states?

food for the slide eater