CSEP590 – Model Checking and Automated Verification

Lecture outline for July 9, 2003

-Formal Verification is composed of 3 steps:
- -1) a framework for modelling the system (last time)
- -2) a specification language to describe properties to be verified
- -3) a verification method to establish if system satisfies specs

-We use a model-based approach. Given a formula ϕ and model M of system, determine if M satisfies ϕ (denoted as $M \models \phi$)

-Specifications written in Temporal Logic
- -formula isn't statically true/false in model
- -dynamic notion of truth
- -classified according to view of time:
  - -linear-time vs. branching time
  - -discrete vs. continuous time

-We will study CTL (computation tree logic) – branching-time + discrete

-CTL formulas are defined inductively in Backus-Naur form (BNF)

-Set of atomic propositions AP, where $p \in AP$

-CTL formula $\phi := \perp \mid T \mid p \mid (\neg \phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid$ $AX\ \phi \mid EX\ \phi \mid A[\phi U\ \phi] \mid E[\phi U\ \phi] \mid AG\ \phi \mid EG\ \phi \mid AF\ \phi \mid EF\ \phi$

-Thus, we have new logical connectives

-AX,EX,AG,EG,AU,EU,AF,EF are *temporal connectives*

-come in pairs: path quantifier + temporal operator

-path quantifiers: A = "along all paths", E = "along some path"

-Temporal operators: X = next state, F = some future state, G = all future states (globally), U = until.

-Ex: EU is actually $E[\phi_1 U\ \phi_2]$. EU and AU are binary operators.

-Notions of well-formed CTL formulas and not well-formed formulas.

-Well-formed include: EGr, AG($q \rightarrow$EGr)…

-Not well-formed include: FGr, EF(rUq), A $\neg$G $\neg$p,…

-Can write out parse trees for well-formed CTL formulas

-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:

-Given a model M of our system, we denote M,s |= ϕ to mean that in state s of M, ϕ holds. Let S denote states of M.

-|= is called satisfaction relation. Defined using structural induction on all CTL formulas:

-1) M,s |= T and M,s |= $\perp$ for all s $\in$S.

-2) M,s |= p iff p $\in$L(s)

-3) M,s |= $\neg\ \phi$ iff M,s !|= ϕ

-4) M,s |= $\phi_1 \wedge \phi_2$ iff M,s |= ϕ_1 and M,s |= ϕ_2

-5) M,s |= $\phi_1 \vee \phi_2$ iff M,s |= ϕ_1 or M,s |= ϕ_2

-6) M,s |= $\phi_1 \rightarrow \phi_2$ iff M,s !|= ϕ_1 or M,s |= ϕ_2

-7) M,s |= AX ϕ iff for all s_1 s.t. $s \rightarrow s_1$ is a transition, we have M,s_1 |= ϕ.

-8) M,s |= EX ϕ iff for some s_1 s.t. $s \rightarrow s_1$ is a transition, we have M,s_1 |= ϕ
-9) M,s |= AG ϕ iff for all paths $s_1 \rightarrow s_2 \rightarrow \ldots$ where s_1 = s, and all for all s_i along the path we have M,s_i |= ϕ
-10) M,s |= EG ϕ iff there exists some path $s_1 \rightarrow s_2 \rightarrow \ldots$ where s_1 = s, and for all s_i along the path we have M,s_i |= ϕ
-11) M,s |= AF ϕ iff for all paths $s_1 \rightarrow s_2 \rightarrow \ldots$ where s_1 = s, there is some s_i on the path s.t. M,s_i |= ϕ
-12) M,s |= EF ϕ iff there exists a path $s_1 \rightarrow s_2 \rightarrow \ldots$ where s_1 = s, there is some s_i on the path s.t. M,s_i |= ϕ
-13) M,s |= A[ϕ_1U ϕ_2] iff for all paths $s_1 \rightarrow s_2 \rightarrow \ldots$ where s_1 = s, the path satisfies ϕ_1U ϕ_2, ie, there is some s_i on the path s.t. M,s_i |= ϕ_2 holds and for each j < i, we have M,s_j |= ϕ_1
-14) M,s |= E[ϕ_1U ϕ_2] iff there is some path $s_1 \rightarrow s_2 \rightarrow \ldots$ where s_1 = s, the path satisfies ϕ_1U ϕ_2, ie, there is some s_i on the path s.t. M,s_i |= ϕ_2 holds and for each j < i, we have M,s_j |= ϕ_1

-What kind of relevant properties can we check with CTL formulas?
    -We'll see some examples in class
-Classical hierarchical classification of verification goals
    -Reachability property – some situation can be reached
        -Ex: "we can enter a critical section"
        -Specified in CTL using the EF operator (EFcrit_sec)
    -Safety property – under certain conditions, something never occurs
        -Ex: "both processes will never be in their critical sections simultaneously"
        -Specified in CTL using the AG, U, or W (weak until) operators
    -Liveness property – under certain conditions, something will ultimately occur
        -Ex: "any request will ultimately be satisfied", "the light will turn green"
        -Tricky to specify, 2 broad families: progress and fairness

-Is liveness even useful? – no bound on notion of when!

-Fairness property – under certain conditions, something will (or will not) occur infinitely often

-Ex: "if access to a critical section is infinitely often requested, then access will be granted infinitely often" – notion of no starvation

-Lots of work in the 1980's. We will discuss it later because it is non-trivial.

-Important equivalences between CTL formulas

-Definition: 2 CTL formulas ϕ and ψ are semantically equivalent if any state in any model which satisfies one of them also satisfies the other. Denoted as $\phi \equiv \psi$.

-We will see some useful ones in lecture

-Equivalences also lead to functionally complete sets for CTL (called adequate sets). One useful set for CTL is $\{AU, EU, EX, \neg, \wedge, \perp\}$

-Now, we develop of model checking algorithm to automatically determine whether $M,s \models \phi$

-Algorithm returns all states s of M which satisfy ϕ

-Routine TRANSLATE(ϕ): pre-processes ϕ to rewrite ϕ in terms of adequate set given above

-Label states of M with subformulas of ϕ satisfied at that state starting with smallest subformulas and working outwards to ϕ

-Suppose ψ is a subformula of ϕ and sates satisfying all immediate subformulas of ψ have been labeled

-Use case analysis to label states with ψ:

-If ψ is:

-$\perp$: no states are labeled with $\perp$

-p: label s with p if $p \in L(s)$

-$\psi_1 \wedge \psi_2$: label s with $\psi_1 \wedge \psi_2$ if s is already labeled with both ψ_1 and ψ_2

-$\neg \psi_1$: label s with ψ_1 if s is not already labeled with ψ_1

-AF ψ_1:
-If any state s is labeled with ψ_1, label it with AF ψ_1
-Repeat: until no change, label any state with AF ψ_1 if all successor states are labeled with AF ψ_1
-E[ψ_1U ψ_2]:
-If any state is labeled with ψ_2, label it with E[ψ_1U ψ_2]
-Repeat: until no change, label any state with E[ψ_1U ψ_2] if it is labeled with ψ_1 and at least 1 successor is labeled with E[ψ_1U ψ_2]
-EX ψ_1: label any state with EX ψ_1 if one of its successors is labeled with ψ_1
-Finally, just output all states labeled with ϕ and we are done!
-Complexity? = O(f*V*(V+E)) where f = # of connectives in ϕ, V = # of states in M, E = # of transitions in M
-=> linear in formula size, quadratic in model size

-Is there a faster way? Yes!
-Handle EG and AG directly:
-EG ψ_1:
-Label all states with EGψ_1
-If any sate is not labelled with ψ_1, delete label EGψ_1
-Repeat: until no change, delete label EGψ_1 from any state if none of its successors are labeled with EGψ_1
-Turns out, there is even a more cleverer way of handling EG (in book). Using adequate set of {EX,EU,EG,$\neg$,$\wedge$,$\perp$} one can achieve a complexity of O(f*(V+E)) => linear in both the size of the formula and the model!
-NEXT LECTURE: briefly touch on other temporal logics (LTL, CTL*), symbolic model checking, fairness, and our first real system: SMV