CS 745 / ECE 725
Computer Aided Verification

Lecture 6: Explicit CTL Model Checking

Jo Atlee

DC 2337, jmatlee@uwaterloo.ca

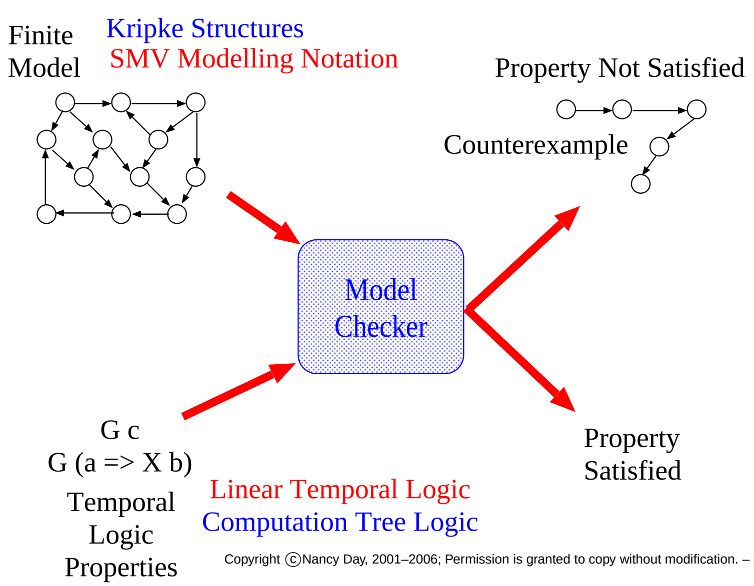
Office Hours: Mon 1:00-2:00, Wed 1:00-2:00

http://www.student.cs.uwaterloo.ca/~cs745

Today's Agenda

- Explict CTL model checking

Model Checking



Roadmap

Problem

Given a model $\mathcal{M}$ (usually a Kripke structure) that represents the behaviour of a system, a state s, and a temporal logic formula f that represents a desired property of the system, determine whether the state in model $\mathcal{M}$ satisfies the formula:

$$\mathcal{M}, s \models f$$

Model Checking Algorithms that we'll study:

- Explicit CTL Model Checking: labelling a graph
- Symbolic CTL Model Checking: representing sets of states using propositional logic

Explicit CTL Model Checking

A Kripke structure $\mathcal{M} = (S, S_0, R, L)$ is a labeled, directed graph.

- Nodes represent the states in S
- Edges represent the transition relation R
- Propositions associated with the nodes are given by the function $L : S \rightarrow 2^{\mathsf{AP}}$.

Introduces another function, label, that labels each state with the temporal formulae that are true of that state

Sources: Clarke, Emerson, Sistla [CES86], Clarke, Grumberg, Peled [CGP99], Huth and Ryan [HR04]

Explicit CTL M/C

The algorithm works recursively over the structure of the formula.

Starting with the atomic propositions that are true in each state

$$\underline{\text{label}}(s) := L(s)$$

And continuing to label states with increasingly more complex subformulae.

Subformulae

The depth of a formula is the height of its parse tree. A depth of 0 means the subformula is a leaf node in the parse tree. A depth of 1 means that the subformula consists of a single connective and one or two operands.

1. Decompose formula f into subformulae
2. $\underline{\text{label}}(s) := L(s)$ labels each state with subformulae of depth 0 that are true in that state.
3. Label every state with all subformulae of depth 1 that are true in the state.

   ⋮

n + 2. Label every state with all subformulae of depth n (which is the formula f) that are true in the state.

Explicit CTL M/C

Test whether $\underline{\text{label}}(s)$ contains formula f

$$\mathcal{M}, s \models f \text{ iff } f \in \underline{\text{label}}(s)$$

To check whether a formula holds of all reachable states, check whether the model's initial states are all labelled with f.

$$\mathcal{M} \models f \quad \text{iff} \quad \forall s_0 \in S_0 \bullet \; s_0 \in \{s \mid \mathcal{M}, s \models f\}$$

Explicit CTL M/C

Want to reduce the number of different types of subformulae to check.

Any CTL formula can be expressed in terms of $\neg, \vee,$ **EX**, **EU**, **EG**:

$f_1 \wedge f_2 = \neg(\neg f_1 \vee \neg f_2)$

EF $f = $ **E**[**true U** f]

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

Explicit CTL M/C

Check $\neg f_1$
  Add $\neg f_1$ to all label(s) if $f_1 \notin$ label(s)

Complexity: $O(|S|)$ (walk over the set of states)

Check $f_1 \vee f_2$
  Add $f_1 \vee f_2$ to label(s) if either f_1 or f_2 are in label(s)

Complexity: $O(|S|)$ (walk over the set of states)

Recall: S is the set of states, R is the transition relation. $|S|$ is the size of the state space.

Explicit CTL M/C: EX

Label every state that has some successor labelled by f_1.

CheckEX(f_1)
  $K = \{s \mid f_1 \in$ label$(s)\}$;
  while $K \neq \emptyset$ do
    choose $s \in K$;
    $K := K \setminus \{s\}$;
    for all $(t, s) \in R$ do
      label(t) := label$(t) \cup$ {**EX**f_1};

Complexity: $O(|S| + |R|)$

Explicit CTL M/C: EU

Find all states that are labelled with f_2. Work backwards using R to find all states that can be reached by some path in which each state is labelled with f_1. Label all these states with **E**[f_1 **U** f_2].

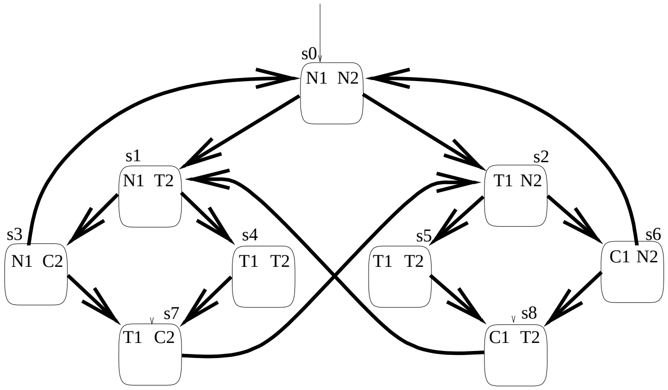
Explicit CTL M/C: EU

CheckEU(f_1, f_2)
$K := \{s \mid f_2 \in$ label(s)$\}$;
for all $s \in K$ do label(s) := label$(s) \cup \{$**E**$[f_1$ **U** $f_2]\}$;
while $K \neq \emptyset$ do
choose $s \in K$;
$K := K \setminus \{s\}$;
for all $(t, s) \in R$ do
if $f_1 \in$ label(t) then
label(t) := label$(t) \cup \{$**E**$[f_1$ **U** $f_2]\}$;
$K := K \cup \{t\}$;

Complexity: $O(|S| + |R|)$

Example: Mutual Exclusion

Two processes that cycle through three states: non-critical (N), trying to enter critical section (T), and critical (C).



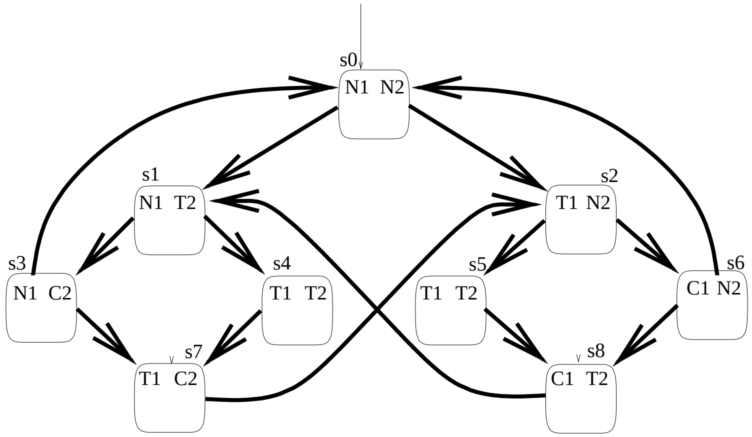
s_4 and s_5 are separate states to prevent starvation.

Properties of Mutual Exclusion

1. (safety) Only one process can be in its critical section at any time.
2. (liveness) Whenever a process wants to enter its critical section, it will eventually be permitted to do so.

Checking Safety

$\neg$**E**[**true U** $\neg(\neg C1 \vee \neg C2)$]



Checking Liveness

Property: Whenever a process wants to enter its critical section, it will eventually be permitted to do so.

$$\textbf{AG}(T1 \Rightarrow \textbf{AF}\ C1)$$

alternatively:

$$\neg(\textbf{E}[\textbf{true U}\ \neg(\neg T1 \vee \neg(\textbf{EG}\ \neg C1))])$$

Explicit CTL M/C: EG (First Try)

EG f_1: Label all states labelled with f_1 with **EG** f_1 Repeat: delete the label **EG** f_1 from any state if none of its successors is labelled with **EG** f_1; until there is no change.

CheckEG(f_1)
  $K := \{s \mid f_1 \in \underline{\text{label}}(s)\}$;
  for all $s \in K$ do $\underline{\text{label}}(s) := \underline{\text{label}}(s) \cup \{\textbf{EG} f_1\}$;
  do
    $K := \{s \mid \textbf{EG} f_1 \in \underline{\text{label}}(s)\}$;
    for all $t \in K$ do
      $\underline{\text{label}}(t) := \underline{\text{label}}(t) - \{\textbf{EG} f_1\}$;
      for all $(t, u) \in R$ do
        if $u \in K$ then $\underline{\text{label}}(t) := \underline{\text{label}}(t) \cup \{\textbf{EG} f_1\}$;
  until $K = \{s \mid \textbf{EG} f_1 \in \underline{\text{label}}(s)\}$;

Explicit CTL M/C: EG

We can do better than this algorithm by noting that for a state to satisfy **EG**f_1, it must have a path leaving it that eventually ends up in a loop in which f_1 is always true.

Strongly Connected Components

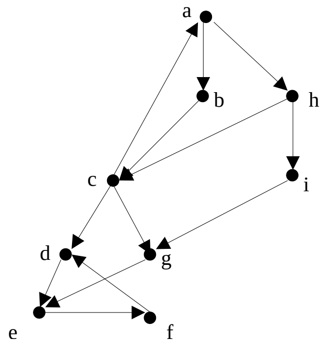
From Manber [Man89]:

A directed graph is strongly connected if, for every pair of vertices v and w, there is a path from v to w. In other words, it is possible to reach any vertex from any other vertex.

A strongly connected component is a maximal subset of the vertices such that its induced subgraph is strongly connected.

A strongly connected component is nontrivial "iff it either has more than one node or it contains one node with a self-loop" [CGP99], p. 36.

Strongly Connected Components



Tarjan's algorithm: $O(|S| + |R|)$

Explicit CTL M/C

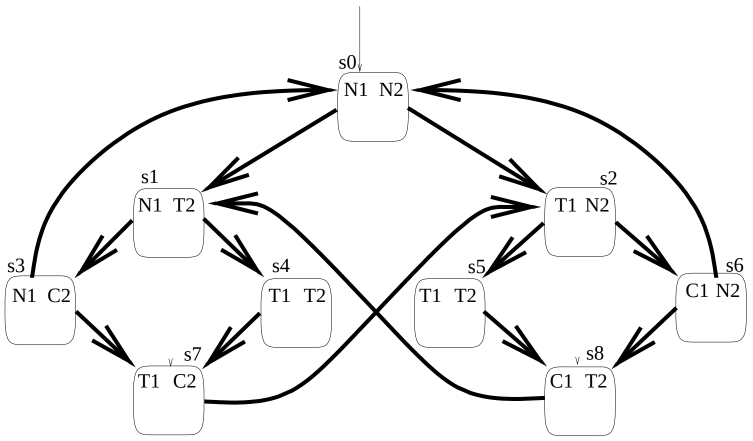
CheckEG(f_1), a more efficient algorithm:

$S' := \{s \mid f_1 \in \underline{\text{label}}(s)\}$;
$SCC := \{C \mid C$ is a nontrivial SCC of $S'\}$;
$K := \bigcup_{C \in SCC} \{s \mid s \in C\}$;
for all $s \in K$ do $\underline{\text{label}}(s) := \underline{\text{label}}(s) \cup \{\textbf{EG} f_1\}$;
while $K \neq \emptyset$ do
choose $s \in K$;
$K := K \setminus \{s\}$;
for all t such that $t \in S'$ and $(t, s) \in R$ do
if $\textbf{EG} f_1 \notin \underline{\text{label}}(t)$ then
$\underline{\text{label}}(t) := \underline{\text{label}}(t) \cup \{\textbf{EG} f_1\}$;
$K := K \cup \{t\}$;

Complexity: $O(|S| + |R|)$

Checking Liveness

$\neg(\textbf{E}[\textbf{true U} \neg(\neg T1 \vee \neg(\textbf{EG} \neg C1))])$



Algorithm Complexity

- Worst case complexity of checking each subformula: $O(|S| + |R|)$
- For a formula of length n, there are n sub-formulae to check.
- Worst case complexity to check a formula of length n is $O(n \times (|S| + |R|))$

Thus, complexity is linear in the size of the formula and in the size of the state space. But the size of the state space is exponential in the number of parallel system components and in the number of variables!

State Space Explosion Problem

References

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic. ACM Transactions on Programming Languages and Systems, 8(2):244–263, April 1986.

[CGP99] Edmund Clarke Jr., Orna Grumberg, and Doron A. Peled. Model Checking. MIT Press, 1999.

[HR04] Michael R. A. Huth and Mark D. Ryan. Logic in Computer Science. Cambridge University Press, Cambridge, 2004. Second Edition.

[Man89] Udi Manber. Introduction to Algorithms: A Creative Approach. Addison-Wesley Publishing Company, Reading , Massachusetts, 1989.

Summary

- Explict CTL model checking

Next class: Symbolic CTL model checking

