Overview

- Idea of model-checking
- State of the art in model-checking
- CTL
- CTL model-checking
- Introduction to BDDs
- Symbolic model-checking
- SMV input language and its semantics
- Notion of fairness
- SMV examples

CTL

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

Syntax:
1. Every atomic proposition is a CTL formula
2. If f and g are CTL formulae, then so are $\sim f$, $f \wedge g$, $f \vee g$, 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?

Fig. 1, page 247 of TOPLASS paper

CTL - Cont'd

Only X and U are necessary. The rest are abbreviations:

$$
\begin{array}{lcl}
(f \wedge g) & \equiv & \sim(\sim f \vee \sim g) \\
(f \rightarrow g) & \equiv & (\sim f \vee g) \\
\mathsf{AX}f & \equiv & \sim\mathsf{EX}(\sim f) \\
\mathsf{EF}f & \equiv & \mathsf{E}[\mathsf{true}\ \mathsf{U}\ f] \\
\mathsf{AF}f & \equiv & \mathsf{A}[\mathsf{true}\ \mathsf{U}\ f] \\
\mathsf{EG}f & \equiv & \sim\mathsf{AF}(\sim f) \\
\mathsf{AG}f & \equiv & \sim\mathsf{EF}(\sim f)
\end{array}
$$

CTL - Cont'd

Formula is defined with respect to a model $M = \langle V, S, s_0, R, I \rangle$, where
- V is a set of propositional variables
- S is a set of states
- $s_0 \in S$ is the start state
- R is a transition relation
- I is a set of interpretations specifying which propositions are true in each state

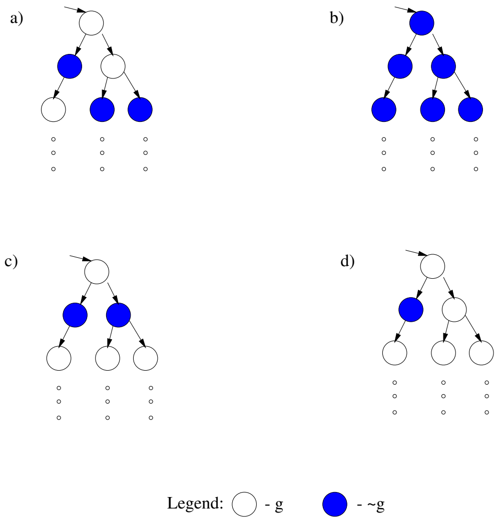
Temporal logic formulas are evaluated with respect to a state in the model:
- EX(f) (AX(f)) is true in s_i if f is true in some (all) successor of s_i
- E[fUg] (A[fUg]) is true in s_i if along some (every) path emanating from s_i there is a future state s_j at which g holds and f is true until state s_j is reached
- EG(f) (AG(f)) is true in s_i if f holds in every state along some (every) path emanating from s_i

Formally...

$$
\begin{array}{rll}
s \models p & \text{iff} & (I(s))(p) \\
s \models \sim f & \text{iff} & s \not\models f \\
s \models f \vee g & \text{iff} & s \models f \text{ or } s \models g \\
s_0 \models \mathsf{EX} \sim f & \text{iff} & \exists \text{ path } (s_0, s_1, \ldots)), s_1 \models \sim f \\
s_0 \models \mathsf{AX} \sim f & \text{iff} & \forall \text{ paths } (s_0, s_1, \ldots)), s_1 \models \sim f \\
s_0 \models \mathsf{E}(f \mathsf{U} g) & \text{iff} & \exists \text{ path } (s_0, s_1, \ldots)), \\
 & & \exists i \text{ s.t. } s_i \models g \text{ and } \forall j < i, s_j \models f \\
s_0 \models \mathsf{A}(f \mathsf{U} g) & \text{iff} & \forall \text{ paths } (s_0, s_1, \ldots)), \\
 & & \exists i, \text{ s.t. } s_i \models g \text{ and } \forall j < i, s_j \models f
\end{array}
$$

Examples

What formulas hold in these models?



Specifications using CTL

Thermostat.

1. Temperature is always above normal, below normal or normal.

2. When temperature becomes above normal, in the following state the AC will be turned on.

3. Thermostat is never running heater and AC at the same time.

CTL Model checking

Assumptions:
1. finite number of processes, each having a finite number of finite-valued variables.
2. finite length of CTL formula

Problem:
Determine whether formula f_0 is true in the finite structure M.

Algorithm overview:
Determine the set of states in which subformulas of f_0 of length 1 hold. Then length 2, etc. until length f_0 is reached. If starting state s_0 is in the final set, then f_0 is holds on M, i.e.

$$(s_0 \in \{s \mid M, s \models f_0\}) \Rightarrow (M \models f_0)$$

It is not as easy as it seems

Formula EFg represents the set of states from which a state satisfying g can be reached in *some* number of transitions:

$$g \vee \mathsf{EX}g \vee \mathsf{EX}(\mathsf{EX}g) \vee \ldots$$

So, we need to do the least fixed-point operation

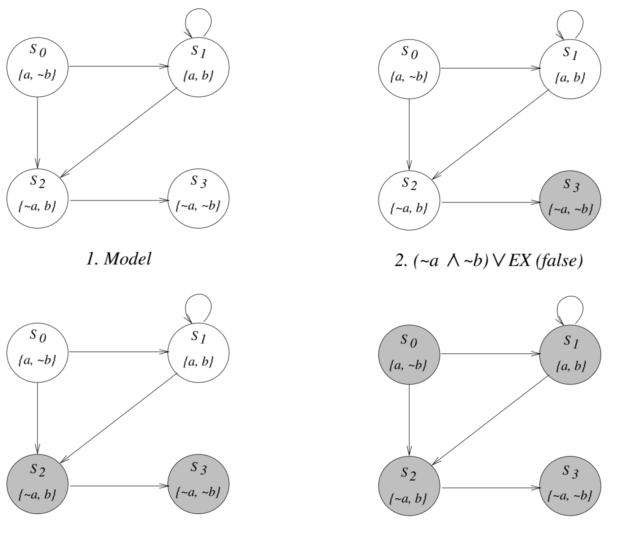
$$\mathsf{EF}g = \mu.y(g \vee \mathsf{EX}y)$$

starting with value *false* for y.

The entire state space of the model must be constructed before the fixed-point algorithms can be applied!!!! Most important problem - state space explosion.

Example

Model checking $s_0 \models \mathsf{EF}(\sim a \wedge \sim b)$.



1. Model

2. (~a ∧ ~b) ∨ EX (false)

3. (~a ∧ ~b) ∨ EX (~a ∧ ~b)

4. (~a ∧ ~b) ∨ EX ((~a ∧ ~b) ∨ EX (~a ∧ ~b))

Symbolic model checking

Why?
Saves is from constructing a model's state space. Effective "cure" for state space explosion problem.

How?
Sets of states and transition relations are represented by formulas, and set operations are defined in terms of formula manipulations.

Data structures
BDDs - allow for efficient storage and manipulation of logic formulas.

Some more detail

- A system state represents an interpretation (truth assignment) for a set of propositional variables V.

- Formulas represent sets of states that satisfy it

a - set of states in which a is true - ($\{s_0, s_1\}$)

b - set of states in which b is true - ($\{s_1, s_2\}$)

$a \vee b = \{s_0, s_1, s_2\}$

- State transitions are described by relations over two sets of variables, V (source state) and $V\prime$ (destination state)

Transition from s_2 to s_3 is described by ($\sim a \wedge b \wedge \sim a\prime \wedge \sim b\prime$).

Transition from s_0 to s_1 and s_2, and from s_1 to s_2 and to itself is described by ($a \wedge b\prime$).

Relation R is described by

$(a \wedge b\prime) \vee (\sim a \wedge b \wedge \sim a\prime \wedge \sim b\prime)$

Symbolic model checking (Cont'd)

The meaning for CTL formulas can be redefined in terms of sets of states:

$$\begin{array}{rll}
s \models f & \text{iff} & s \in f \text{ where } f \in V \\
s \models \sim f & \text{iff} & s \in \neg f \\
s \models f \vee g & \text{iff} & s \in (f \vee g) \\
s \models \mathsf{EX} f & \text{iff} & s \in (\exists V'(R \wedge f(V/V'))) \\
s \models \mathsf{AX} f & \text{iff} & s \in \sim (\exists V'(R \wedge \sim f(V/V'))) \\
s \models \mathsf{E}(f \mathsf{U} g) & \text{iff} & s \in \mu y.(g \vee (f \wedge \mathsf{EX} \sim y)) \\
s \models \mathsf{A}(f \mathsf{U} g) & \text{iff} & s \in \mu y.(g \vee (f \wedge \mathsf{AX} y))
\end{array}$$

Symbolic model checking

- A CTL formula f is evaluated for a model by deriving a *propositional logic expression* that describes the set of states satisfying the CTL formula for the model.
- The model checker verifies that the interpretation of the model's initial state s_0 satisfies the expression.

Example - check $s_0 \models \mathsf{EX}(\sim a \wedge \sim b)$, i.e., compute a formula representing the states that have successors where $(\sim a \wedge \sim b)$ is true:
- R - the transition relation
- f - the formula being checked
- $f(a, b/a\prime, b\prime)$ - substitution, leading to reasoning about the next state
- Replace formulas like $\exists v, (f)$ by $(f(v/true) \vee f(v/false))$.

Computation

$$\begin{array}{l}
EX(\sim a \wedge \sim b) = \\
\exists a', b'(((a \wedge b') \vee (\sim a \wedge b \wedge \sim a' \wedge \sim b')) \\
\quad \wedge ((\sim a \wedge \sim b)(a, b/a', b'))) = \\
\exists a', b'(((a \wedge b') \vee (\sim a \wedge b \wedge \sim a' \wedge \sim b')) \\
\quad \wedge (\sim a' \wedge \sim b')) = \\
\exists a', b'((a \wedge b' \wedge \sim a' \wedge \sim b') \\
\quad \vee (\sim a \wedge b \wedge \sim a' \wedge \sim b' \wedge \sim a' \wedge \sim b')) = \\
\exists a', b'((false) \vee (\sim a \wedge b \wedge \sim a' \wedge \sim b')) = \\
\exists a', b'(\sim a \wedge b \wedge \sim a' \wedge \sim b') = \\
\exists a'((\sim a \wedge b \wedge \sim a' \wedge \sim true) \vee \\
\quad (\sim a \wedge b \wedge \sim a' \wedge \sim false)) = \\
\exists a'(\sim a \wedge b \wedge \sim a') = (\sim a \wedge b \wedge \sim true) \\
\quad \vee (\sim a \wedge b \wedge \sim false) = \sim a \wedge b
\end{array}$$

Evaluation of example

The computed propositional logic formula represents the set of states whose interpretations satisfy EX($\sim a \wedge \sim b$), that is, $\{s_2\}$.

EX($\sim a \wedge \sim b$) is a theorem (i.e., $s_0 \models$ EX($\sim a \wedge \sim b$)) if the values of a and b in s_0 satisfy $\sim a \wedge b$.

In s_0, a=true, b=false. So, $s_0 \not\models \sim a \wedge b$. Thus, $s_0 \not\models$ EX($\sim a \wedge \sim b$).

Symbolic model checking

Example: calculate EF($\sim a \wedge \sim b$) for our model:

1) $(\sim a \wedge \sim b) \vee EX false = (\sim a \wedge \sim b)$
2) $(\sim a \wedge \sim b) \vee EX(\sim a \wedge \sim b) = (\sim a \wedge \sim b) \vee (\sim a \wedge b) = \sim a$
3) $(\sim a \wedge \sim b) \vee EX(\sim a) =$

$(\sim a \wedge \sim b) \vee \exists a', b'(((a \wedge b') \vee (\sim a \wedge b \wedge \sim a' \wedge \sim b'))$
$\quad \wedge((\sim a)(a/a'))) =$
$(\sim a \wedge \sim b) \vee \exists a', b'(((a \wedge b') \vee (\sim a \wedge b \wedge \sim a' \wedge \sim b')) \wedge \sim a') =$
$(\sim a \wedge \sim b) \vee \exists a', b'((a \wedge b' \wedge \sim a') \vee (\sim a \wedge b \wedge \sim a' \wedge \sim b')) =$
$(\sim a \wedge \sim b) \vee \exists a'(((a \wedge true \wedge \sim a') \vee (\sim a \wedge b \wedge \sim a' \wedge \sim true))$
$\quad \vee((a \wedge false \wedge \sim a') \vee (\sim a \wedge b \wedge \sim a' \wedge \sim false))) =$
$(\sim a \wedge \sim b) \vee \exists a'((a \wedge \sim a') \vee (\sim a \wedge b \wedge \sim a')) =$
$(\sim a \wedge \sim b) \vee ((a \wedge \sim true)$
$\quad \vee(\sim a \wedge b \wedge \sim true)) \vee ((a \wedge \sim false) \vee (\sim a \wedge b \wedge \sim false)) =$
$(\sim a \wedge \sim b) \vee ((false) \vee (false)) \vee ((a) \vee (\sim a \wedge b)) =$
$(\sim a \wedge \sim b) \vee (\sim a \wedge b) \vee a = true$

4) $(\sim a \wedge \sim b) \vee EX(true) = true$

Symbolic model checking

Note that the formulas at the end of each step correspond to the set of states that is shaded after that step.

- The first formula, ($\sim a \wedge \sim b$), corresponds to $\{s_3\}$
- The second formula, ($\sim a$), corresponds to $\{s_2, s_3\}$
- The last formula, *true*, corresponds to the set of all states.

food for slide eater