

Notes on Model Checking

Bud Mishra^{1,2}

¹ Courant Institute of Mathematical Sciences, NYU

² Watson School of Biological Sciences, CSHL

Courant/NYU Bioinformatics Group

April 28, 2003

Introduction

Model Checking & Kripke Structure

- ◇ Definition: *Kripke Structure*
- ◇ ...captures the intuition about behavior of a reactive system...
- ◇ ...consists of a set of states, a set of transitions between states, **and a function that labels each state with a state of properties — true in that state.**

- ◇ Formal Definition Let AP be a set of atomic propositions. A *Kripke Structure* M over AP is a four tuple $M = (S, S_0, R, L)$ where
- S is a finite set of states.
 - $S_0 \subseteq S$ is a set of initial states.
 - $R \subseteq S \times S$ is a transition relation that must be total, that is, for every state $s \in S$ there is a state $s' \in S$ such that $R(s, s')$.
 - $L : S \leftarrow 2^{AP}$ is a function that labels each state with the state of atomic propositions that are true in that state.

First Order Representation

- ◇ **Logical Conectives:** and \wedge , or \vee , not \neg , implies \rightarrow , so on.
- ◇ **Quantifiers:** universal quantifiers \forall , existential quantifiers \exists , so on.
- ◇ A logic is propositional, if it consists of atomic propositions and formulas created with the logical connectives... but no quantifiers.
- ◇ A logic is first order if the atomic propositions take values in a domain (not necessarily finite) and in addition the formulas are quantified over the domain.

Example



$$\begin{aligned} &\forall \text{initial cell mass, } c, k_1 < c < k_2 [CellIn(S, 0) \rightarrow \forall_{t>0} CellIn(S, t)] \\ &\wedge \forall \text{initial cell mass, } c, c > k_2 [CellIn(S, 0) \rightarrow \exists_{t>0} CellIn(M, t)] \end{aligned}$$

Temporal Logic

- ◇ Temporal Logic is a formalism for describing a sequence of transitions between states in a reactive system...
- ◇ In this logic, time is never mentioned explicitly with a metric... but in a “topological” manner

- ◇ “Eventually something happens” ... “Always something is true” ... “Something is never true” ... “Something else holds almost always” ... “This is true infinitely often” ...

- ◇ Main **modes** or temporal operators are \mathcal{X} , \mathcal{F} , \mathcal{G} , \mathcal{U} and \mathcal{R} .

- ◇ Main **path quantifiers** are \mathcal{A} and \mathcal{E} .

Computation Tree Logic CTL

- ◇ 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...
- ◇ The tree shows all possible execution paths in the tree...
- ◇ CTL formulas are composed of path quantifiers and temporal operators.

Computation Tree Logic CTL

- ◇ Next Time: $X P \dots$ property P holds in the second state of the path.
- ◇ Eventually: $F P \dots$ property P will hold at some state on the path (in the future)
- ◇ Always: $G P \dots$ property P holds at every state on the path (globally)
- ◇ Until: $P U Q \dots$ property Q holds at some state on the path and property P holds at all preceding states
- ◇ Release: $P R Q \dots$ property Q holds along the path up to and including the first state at which P holds (if it does at all)

Computation Tree Logic CTL

- ◇ There are two types of formulas in CTL: *state formulas* and *path formulas*... state formulas are true in a specific state... path formulas are true along a specific path...

- ◇ Let AP be the set of atomic proposition names.

- ◇ The syntax of state formulas:
 - If $p \in AP$, then p is a state formula.
 - If f and g are state formulas, then $\neg f$, $f \vee g$ and $f \wedge g$ are state formulas.
 - If f is a path formula then $\mathcal{E} f$ and $\mathcal{A} f$ are state formulas.

◇ The syntax of path formulas:

- If f is a state formula, then f is also a path formula.
- If f and g are path formulas, then $\neg f$, $f \vee g$, $f \wedge g$, $\mathcal{X}f$, $\mathcal{F}f$, $\mathcal{G}f$, $f\mathcal{U}g$ and $f\mathcal{R}g$ are path formulas.

Semantics of CTL with Kripke Structure

- ◇ $M = \langle S, R, L \rangle$ = a Kripke Structure. S = the set of states, $R \subseteq S \times S$ = transition relation (total) and $L: S \rightarrow 2^{AP}$ = the labeling function... labels each state with a set of atomic propositions true in that state
- ◇ A path in M is an infinite sequence of states $\pi = s_0, s_1, \dots$ such that $\forall_{i \geq 0} (s_i, s_{i+1}) \in R$

Semantics of CTL with Kripke Structure

- ◇ $M, s \models p$ iff $p \in L(s)$
- ◇ $M, s \models \neg f_1$ iff $M, s \not\models f_1$
- ◇ $M, s \models f_1 \vee f_2$ iff $M, s \models f_1$ or $M, s \models f_2$
- ◇ $M, s \models f_1 \wedge f_2$ iff $M, s \models f_1$ and $M, s \models f_2$
- ◇ $M, s \models \mathcal{E}g_1$ iff there is a path π from s s.t. $M, \pi \models g_1$
- ◇ $M, s \models \mathcal{A}g_1$ iff for every path π from s s.t. $M, \pi \models g_1$
- ◇ $M, \pi \models f_1$ iff s is the first state of path π and $M, s \models f_1$
- ◇ $M, \pi \models \neg g_1$ iff $M, \pi \not\models g_1$

◇ $M, \pi \models g_1 \vee g_2$ iff $M, \pi \models g_1$ or $M, \pi \models g_2$

◇ $M, \pi \models g_1 \wedge g_2$ iff $M, \pi \models g_1$ and $M, \pi \models g_2$

◇ $M, \pi \models \mathcal{X}g_1$ iff $M, \pi^1 \models g_1$

◇ $M, \pi \models \mathcal{F}g_1$ iff $\exists k \geq 0$ s.t. $M, \pi^k \models g_1$

◇ $M, \pi \models \mathcal{G}g_1$ iff $\forall k \geq 0, M, \pi^k \models g_1$

◇ $M, \pi \models g_1 \mathcal{U} g_2$ iff $\exists k \geq 0$ s.t. $M, \pi^k \models g_2$ and $\forall 0 \leq j < k, M, \pi^j \models g_1$

◇ $M, \pi \models g_1 \mathcal{R} g_2$ iff $\forall k \geq 0, \text{ if } \forall 0 \leq j < k M, \pi^j \not\models g_1 \text{ then } M, \pi^k \models g_2$

Least Fixed Point Characterization

◇ It suffices to define all path formulas in terms of: P , $\neg f$, $f_1 \wedge f_2$, $\mathcal{A}\mathcal{X}f_1$, $\mathcal{E}\mathcal{X}f_1$, $\mathcal{A}[f_1\mathcal{U}f_2]$ and $\mathcal{E}[f_1\mathcal{U}f_2]$

$$\diamond P \equiv \mu z.P$$

$$\diamond \neg f_1 \equiv \mu z.\neg f_1$$

$$\diamond f_1 \wedge f_2 \equiv \mu z.f_1 \wedge f_2$$

$$\diamond \mathcal{A}\mathcal{X}f_1 \equiv \mu z.\mathcal{A}\mathcal{X}f_1$$

$$\diamond \mathcal{E}\mathcal{X}f_1 \equiv \mu z.\mathcal{E}\mathcal{X}f_1$$

$$\diamond \mathcal{A}[f_1 \mathcal{U} f_2] \equiv \mu z. f_2 \vee (f_1 \wedge \mathcal{A}\mathcal{X}z)$$

$$\diamond \mathcal{E}[f_1 \mathcal{U} f_2] \equiv \mu z. f_2 \vee (f_1 \wedge \mathcal{E}\mathcal{X}z)$$

Algorithm

Label-Graph(f, M)

begin case:

- ◇ $f = P$:
while $\exists s \in S$ s.t. [$f \notin Lbl(s)$ and $P \in L(s)$]
Add f to $Lbl(s)$

- ◇ $f = \neg f_1$:
Label-Graph(f_1, M);
while $\exists s \in S$ s.t. [$f \notin Lbl(s)$ and $f_1 \notin Lbl(s)$]
Add f to $Lbl(s)$

- ◇ $f = f_1 \wedge f_2$:
Label-Graph(f_1, M);
Label-Graph(f_2, M);
while $\exists s \in S$ s.t. [$f \notin Lbl(s)$ and $f_1 \in Lbl(s)$ and $f_2 \in Lbl(s)$]
Add f to $Lbl(s)$

- ◇ $f = \mathcal{A}\mathcal{X}f_1$:
Label-Graph(f_1, M);

while $\exists s \in S$ s.t. [$f \notin Lbl(s)$ and $\forall t \in succ(s), f_1 \in Lbl(t)$]
Add f to $Lbl(s)$

◇ $f = \mathcal{E}\mathcal{X}f_1$:
Label-Graph(f_1, M);
while $\exists s \in S$ s.t. [$f \notin Lbl(s)$ and $\exists t \in succ(s), f_1 \in Lbl(t)$]
Add f to $Lbl(s)$

◇ $f = \mathcal{A}[f_1\mathcal{U}f_2]$:
Label-Graph(f_1, M);
Label-Graph(f_2, M);
while $\exists s \in S$ s.t. [$f \notin Lbl(s)$ and ($f_2 \in Lbl(s)$
or ($f_1 \in Lbl(s)$ and $\forall t \in succ(s), f \in Lbl(t)$)))]
Add f to $Lbl(s)$

◇ $f = \mathcal{A}[f_1\mathcal{U}f_2]$:
Label-Graph(f_1, M);
Label-Graph(f_2, M);
while $\exists s \in S$ s.t. [$f \notin Lbl(s)$ and ($f_2 \in Lbl(s)$
or ($f_1 \in Lbl(s)$ and $\exists t \in succ(s), f \in Lbl(t)$)))]
Add f to $Lbl(s)$

The End