6.894 : Lightweight Formal Methods

Lecture 11 : Temporal Logic I

Daniel Jackson, March 28, 2005.

Why Model checking?

We've seen how Alloy gives limited ability to reason about temporal behaviour: only on bounded traces, typically short.

Model checking allows analysis of all traces of a finite state machine. Modern model checkers can scale to machines of $> 10^{60}$ states, depending on how uniform their structure is.

Instead of arbitrary temporal properties expressed in FOL, they use special temporal logics. For example, LTL, which has 'always' and 'eventually' operators - less expressive than FOL but more tractable.

Aside from arb. trace length, other major difference from Alloy is lack of data structures. Most model checking languages allow only v. low-level data structures.

Basic form of model checking query:

$$M, s \models P$$

- M: machine, expressed as a set of guarded commands
- s: state
- P: temporal logic formula

Does machine M satisfy property P in state s?

History of Model checking

1975 Dijkstra's Guarded Commands
1976-9 Jan Hajek's "Approver"
1977 Pnueli's Temporal Logic.
1978 Hoare's CSP.
1981 Clarke + Emerson
Queille + Sifakis
'model checking'
1980 Holzmann's PAN
1986 Wolper + Vardi: automaton framework for LTL.
1989 SPIN version 0
1989 McMillan's model checking with OBDD's.
1992 SMV.
1994 Pentium FPU bug: formal verification takes off.

Some current model checkers.

Explicit state.

SPIN

Java Pathfinder

~~Concurrency~~

TLC.

Process algebra.

Concurrency workbench

FDR

LTSA

Symbolic.

SMV : CMU, Cadence.

NUSMV

Specialized: for hybrid systems

HyTech

Uppaal.

Packaged.

SLAM / ~~BLAST~~ SDV Microsoft : now SDV, for Windows drivers.

BLAST Berkeley, MC for C programs

Axes of classification.

Spec language ?

Temporal logic : CTL, LTL. $\Box$ Req $\Rightarrow \Diamond$ Resp

Process algebra $S = Req \rightarrow Resp \rightarrow S$

Machine language

Guarded commands.

C-like syntax, ~~either~~ SPIN

Raw assignments SMV

Synchronous SMV

Asynchronous SPIN.

Analysis method

explicit state SPIN data structures states

language inclusion COSPAN

symbolic SMV, Ahrq. word formula

Applications

Hybrid

Real time

Hardware

Protocols

Software.

Modal Logics.

Logics of necessity and possibility.

- $\Box$ It is necessary that ...
- $\Diamond$ It is possible that ...

Temporal logics

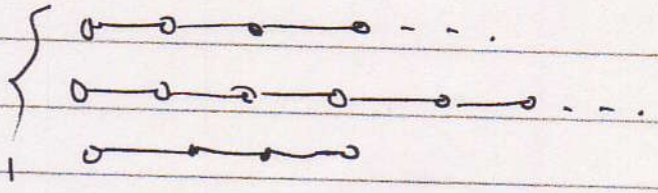
- G It will always be the case that ...
- F It will be the case that ...
- H It has always been the case that ...
- P It was the case that ...

Modal logic invented by Lewis in 1918 to overcome paradox of implication ("false $\Rightarrow$ X" is true) by distinguishing necessary from contingent truth.

Applications in CS originated with Amir Pnueli's work in 1977, applying temporal logic to reasoning about programs.

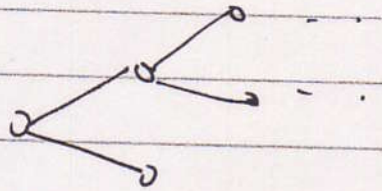
Semantic models for temporal logic

Linear:
(eg. LTL)



set of traces : state sequences

Branching:
(eg. CTL)



tree of states

We'll start by explaining CTL*, a branching-time logic, then we'll explain CTL and LTL as subsets of it.

Computation trees.

Take state machine M with transition relation R over states S

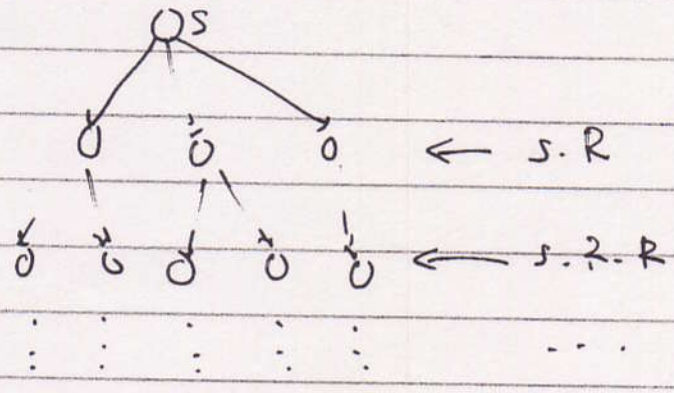
Important assumption:

R is total i.e. S in dom(R)

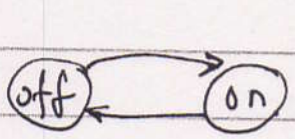
all s: S | some s.R

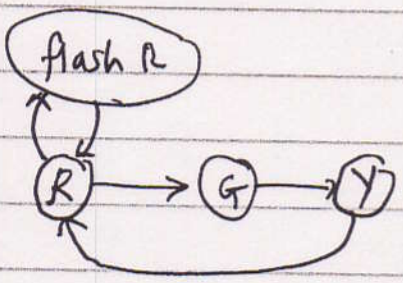
Usually R is not functional : non-determinism.

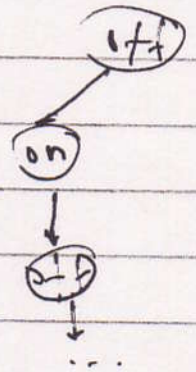
Computation tree T for R, rooted in state s:

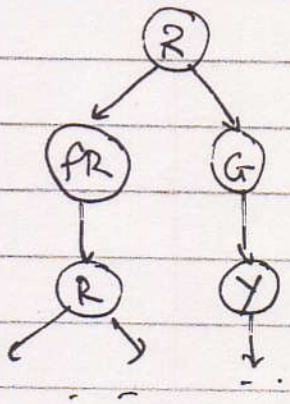


Eg.









Kripke structures-

We'll want to have a notion of evaluation of non-temporal properties of states. Simplest theoretical model — associate set of propositions with each state

Kripke str is $\langle S, R, L \rangle$

$R \subseteq S \times S$

$L : S \to 2^{AP}$

Paths

path or trace $\pi = s_0 s_1 s_2 \ldots$

s.t. $(s_i, s_{i+1}) \in R$

$s_i \to s_{i+1}$ in R

π^i is suffix starting at s_i: $s_i s_{i+1} \ldots$

Rework traffic light.

propositions

R, G, Y, F(lashing)

CTL*

Two kinds of formulas:

state formulas — meaning def'd wrt. a state

path formulas — path.

Syntax.

proposition p

p is a state formula

if f is a

state formulas

$\phi ::= p \mid \text{true} \mid \text{false} \mid \neg\phi \mid \phi \vee \phi \mid \phi \wedge \phi \mid A\alpha \mid E\alpha$

A — always / all paths; E — exists path.

path formulas

$\alpha ::= \phi \mid \neg\alpha \mid \alpha \vee \alpha \mid \alpha \wedge \alpha \mid \alpha \,U\, \alpha$ — until

$G\alpha$ — globally

$F\alpha$ — finally

$X\alpha$ — next

Note that

state formula $\subseteq$ path formula,

but not v.v.

Semantics of CTL*

$M, s \models p \iff p \in L(s)$

$\models \neg f \iff M, s \not\models f$

$\models f \vee g$

$M, s \models E f \quad \exists \pi = s \ldots \mid M, \pi \models f$

$A f \quad \forall \pi = s \ldots \quad M, \pi \models f$

state formula

$M, \pi \models f \iff \pi = s_0 \ldots \quad M, s_0 \models f$

$M, \pi \models X f \iff M, \pi^1 \models f$

$F f \quad \exists k \geq 0 \quad M, \pi^k \models f$

$G f \quad \forall k \geq 0$

$f U g \quad \exists k \geq 0 \; M, \pi^k \models g$

$\wedge \; \forall 0 \leq j < k . \; M, \pi^j \models f$

$\vee, \neg, X, U, E$

are sufficient

$Ff \equiv \text{true } U f$

$Gf \equiv \neg F \neg f$

$Af \equiv \neg E \neg f$