MODEL CHECKING:
Algorithmic Verification II

E. Allen Emerson

Computer Science Department
University of Texas at Austin

Summer School on Model Checking,
Beijing, October 2010

The Meaning(s) of Model Checking

- Original: MC_0 :: auto. ver'n finite systems w/ TL
  Given finite pgm state graph M, TL spec f does $M \models f$?

- Logical: MC_1 :: Given interpretation M, specification f, check if M is a model of f
  Note: suggested the name
  Note: Tarskian def. of truth

- MC_2: Algorithmic Verification is sometimes identified with Model Checking

- MC_3: The general verification problem, decidable or not

- MC_4: LTL model checking: $M \models h$: $Lang(M) \subseteq Lang(H)$: $f_M \Rightarrow h$ is valid
  Note: MC_4 is Not an instance of MC_1.

Theory into Practice: Separation of Concerns Helpful!

- Theory that doesn't work: Useless
- Ideal Cav paper: monolithic, combines:
  - new theory and experimental confirmation of utility
  - Combination slows workflow
  - Decomposing theory & practice speeds workflow
  - Allows more thorough experimentation and theory
- Physics: a more mature discipline:
  - theoretical physics
  - experimental physics
  - asynchronous development of parallel tracks
  - connections b/ theories studied
  - decoupling theory & exper.: faster progress

What to Verify

- Conventional Correctness: against stupidity: hard problem

- “Against stupidity the Gods themselves contend in vain”
  — Friedrich von Schiller

- Security: against malevalence: harder problem
  — Good vs. Evil

- Enron, Houston, Tx; Bernie Madoff, NY, NY
  — Machiavellian, Malevolent

Ultimate Goal: Program Well

- de-**bug**-ging, testing inadequate

- new progr. lang. may help to think about programming,

- but not ensure correctness

- need mathematics

Classic formal verification

- proofs difficult
- require human ingenuity: loop invariants, etc.
- excess clerical, technical detail
- academic success: toy pgms, soundness, completeness
- not scale to industrial systems

There ought to be an easier way

- **Use modal tense logic; temporal logic (TL) [Pn77]** well-suited to reasoning about ongoing, concurrent programs, e.g., OS's, protocols, onboard avionics,... reactive systems
- **Theory**: Any consistent TL spec f realizable in "small" finite state graph model M
- **Practice**: Many concurrent programs are finite state
- **Model Checking**: Given any finite M and spec f **check** that M is a genuine **model** of f: $M \models f$.

Pre-history of Temporal Reasoning

- timing properties: suggest < (before) and > (after)
- Hans Kamp (1968): TL = FOLLO, 1st ord. logic of <, >
- Buchi (1960): S/FOLLO = = finite automata on inf strings
- McNaughton, Pappert (1970): elaborated on long fin. strings
- TL has finite model property:
  - by "Pumping Lemma" for corresponding automaton

TL for concurrency

- Concrrent systems

- Better name?: Reactive Systems

- among subcomponents
  - vis-a-vis environment
  - use Temporal Logic for Specs

Temporal Logic

- formalism for describing change over time

- linear time operators along paths
  - F, G, X, U

- path quantifers for branching time
  - A, E

Common Temporal Logics

- LTL: bool. comb., nestings of F, G, X, U applied to prop. P

- CTL: bool. comb., nestings of (A or E) (F, G, X, or U) applied to prop's.

- CTL*: essentially arb. comb., nestings A,E plus F,G,X,U

Many Temporal Logics and Formalisms

- Branching: CTL, (AFp vs EFp) CTL*, FairCTL
  necessary vs possible; intricacy
- Linear: LTL ($G(p \Rightarrow Fq)$), (ω-)regular expressions
  implicit all paths; simplicity
- Foundational & Practical: Mu-calculus ($\mu Z.P \vee AXZ$)
  Theory of Every Thing; Underlies many tools
- Industrial: IBM Sugar, Accellera/IEEE-1850 PSL, Intel ForSpec
  Tailored for HW with many special "macros"

Preliminaries

Kripke Structure

- $M = (S, R, L)$
  - S state "space",
  - R total, binary transition relation,
  - L labels each state with atomic proposition symbols

- states: system snapshots

- transitions: actions

- $L(s)$: true facts/conditions

Computation Tree Logic (CTL)

- A,E: path quantifiers

- F,G,X,U: linear temporal operators

- basic modalities:

  (A or E) (F, G, X , or U)

- nestings, bool. comb.

CTL examples

- AFp
- EFp
- AGp
- $AGEFp$
- $AG(Try_1 \Rightarrow AFCs_1)$

CTL semantics

- $M, s_0 \models EFp$ iff
  - $\exists$ maximal path $x = s_0, s_1, s_2, \ldots$ $\exists$ time i
  - such that $M, s_i \models p$

- $M, s_0 \models AFp$ iff
  - $\forall$ maximal paths $x = s_0, s_1, s_2, \ldots$ $\exists$ time i
  - such that $M, s_i \models p$

- etc.

Linear Temporal Logic (LTL)

- basic modalities: F, G, X, U
- sometime, always, nexttime, until (resp.)
- Examples:
- $G(\neg(C_1 \wedge C_2))$: mut excl.
- $G(T_1 \Rightarrow FC_1)$: starvation freedom
- semantics: (maximal) paths
- degenerate CTL: elide A, E

Computation Tree Logic* (CTL*)

- subsumes CTL and LTL
- Basic modalities Ah, Eh
- LTL formula h
- nestings, bool. comb.
- $EGFp$: p occurs i.o. (infinitely often) along some path
- $AGFp$: p occurs i.o. along all paths
- $AGFp = AGAFp$, so is expr. in CTL
- $EGFp$ =?= $EGEFp$:
  - one path w/ p i.o. vs comb-like struct.
  - one path with p dangling off each node

BT vs LT (Branching vs Linear Time)

- CTL* strictly subsumes LTL
- in LTL write h, meaning Ah of CTL*
- $M \models h$ means $M \models Ah$
- But (a) " $M \models h$ is false" =/=
  - (b) "$M \models \neg h$"
- equiv. would yield absurd $M \models A\neg h$
- (a) cannot be captured in LTL, due to
- implicit universal path quantification convention
- but can in CTL*
- need $M \models E\neg h$

- LTL is not closed under (semantic) negation

BTvsLT: Show Up

- In “BTvsLT”: Final Showdown” claimed
- BT rendered unusable by
  - tricky, even inscrutable formulae
- consider: $AFAXp =?\ AXAFp$? True/False?
- as given, is confusing
- resolution: exploit “power” of BT
  - have both A and E
- Dualize: - $EGEXp =?\ EXEGp$
- LHS: inf. path w/ p dangling off
  - RHS: inf. path w/ p true almost always (“co-linear”)

- plainly different
- not equiv.

- : Moral: E is often easier to think about than A

Expressiveness Revisited

- reduce $M \models f$ to $M' \models f'$
  - for some derived M',f'
- e.g., EGF to EF
- quadratic blowup
- see next

CTL Model Checking

- $AFp = p \vee AX(AFp)$: is a fixed point of
- $\tau(Z) = Z$ where $\tau(Z) = p \vee AX(Z)$
- $AFp =$ the least fixed point of $Z = \tau(Z)$, denoted $\mu Z.\tau(Z)$, a Mu-calculus expr.
- $AF^{\leq 1}p = p$
- $AF^{\leq i+1}p = p \vee AX(AF^{\leq i}p)$
- $AF^{\leq i}p$ form an ascending chain

- union = uppermost element
- Must stabilize by $i = \text{card}(S)$

LTL Model Checking:I

- Given h build tableau T_h
- View as an automaton Aut_h (cf. [ES83], [VWS83])
- View M as automaton
- Product automaton: $M \times Aut_h$
- Check nonemptiness

LTL Model Checking:II

- use mu-calculus
- LTL h to ω-reg. expr h'.
- LTL Eh to PDL-Δ
- on to Mu-calc.

Automata

- Finite automata on infinite strings, Buchi 1960

- $\mathcal{A} = (Q, \Sigma, \delta, q_0, \Phi)$

- Φ:: acceptance condition

- Buchi acceptance: *green* i.o.

- written in LTL: $GF green$

Other acceptance conditions

- pairs (McNaughton-Rabin):
  some i: *red*$_i$ f.o. and *green*$_i$ i.o

- complemented pairs (Streett)

- parity:: highest *Color*$_i$ flash i.o. has i even

- Muller:: a particular set of states recur i.o.

Nonemptiness

- a **run** r along input string $x = x_1, x_2, x_3 \ldots$
  - is a seq. of automaton states $q_0, q_1, q_2, \ldots$ where
  -$q_0 \rightarrow x_1 \rightarrow q_1 \rightarrow x_2 \rightarrow q_2 ...$
  - is a path through the diagram of $\mathcal{A}$

- $\mathcal{A}$ **accepts** string x iff $\exists$ run r of automaton along x such that $r \models \Phi$

- **Nonemptiness**: does $\mathcal{A}$ accept some string?

Deciding Nonemptiness

- Algorithm(s)

- erase input symbols (a, b, etc.), leave *green*, etc.

- Algorithm I:
  - Calc. states reachable from q_0
  - in induced subgraph, calc. SCCs - nonempty iff exists (nontrivial) SCC C with
  - *green* s in C

- Algorithm II:
  - use mu-calculus

- viewing diagram of $\mathcal{A}$ as a structure
- model check: $\mathcal{A}, q_0 \models EGF green$ thusly
- eval. $\nu Z.EXEF(green \wedge Z)$

Automata for Basic LTL Modalities

- For each of Fp, Gp, GFp, FGp
  - there is a Buchi automaton w/ 2 states
  - 1st three: deterministic; last: nondeter.

- Fp:: start state s_0, green state s_1
  -alphabet $\Sigma = \{p, \neg p\}$
  - s_0, p enter s_0, /* spin */
  $-s_0, p$ enter s_1 /* flash green */
  -s_1 on p or $\neg p$ enter s_1 /* trapped flashing green */

- Gp:: state s_0, start, green
  In state s_0 on p re-enter s_0, flash *green*;

In state s_0 on $\neg p$ enter s_1;
In s_1 on any input re-enter s_1 but no flash.

- GFp:: s_0 start state; s_1 green state;
  On input p from s_0, s_1 enter s_1, flash green;
  On input $\neg p$ from s_0, s_1 enter s_0, not flash

- FGp:: s_0 start, s_1 green state;
  In s_0 consume all input until (guessed)
  time when all $\neg p$'s seen; nondeter. choose to enter s_1
  In s_1 on p flash green
  In s_1 on $\neg p$, abort

- **Exercise**: Prove automata correct.