Model Checking: An Overview

Ahmed Bouajjani

LIAFA, University Paris Diderot – Paris 7

Turunç, June 25, 2012

Model Checking

What is the problem ?

- System/Program $\rightarrow$ Model (state machine)
- Specification/Property $=$ Set of behaviors
- Specification $\rightarrow$ Formula (temporal logic)
- Problem: Model *satisfies* Formula

Issues:

- What kind of models for what kind of systems ?
- What kind of logics for what kind of properties ?
- Decidability ? Complexity ?
- Efficiency, scalability ?
- Under/Upper approximations ?

Models = Various Classes of Automata

After some abstraction ...

- Finite-state automata

  Hardware, communication protocols, etc.

- FSA + stack = pushdown systems

  Boolean procedural programs

- FSA + clocks = timed automata

  Real-time systems

- FSA + counters = counter automata, vector addition syst. (Petri nets)

  Mutual exclusion protocols, cache coherence protocols, device drivers, etc.

- FSA + fifo queues = fifo channel automata

  Communication protocols, distributed systems, etc.

Properties: Behaviors

A behavioral property talks about (infinite) computations.

- Safety / Invariance properties

$$Init \Rightarrow \Box Safe$$

- Termination / Liveness properties

$$\Box Init \Rightarrow \Diamond Termination$$

$$\Box Request \Rightarrow \Diamond Response$$

$$\Box\Diamond Query \Rightarrow \Box\Diamond Grant$$

Specification languages: Temporal Logics (and others ...)

LTL [Pnueli 77], CTL [Clarke, Emerson 82], ...

Properties: States

State properties talks about configurations and relations between configurations (e.g., Input and Ouput of a procedure).

- Specifying states/configurations: FO logic over data domains
- Data domain D (integers, reals, words, terms, ...)
- Program variables $X = \{x_1, \ldots, x_n\}$ over D
- Specification logic: $\mathrm{FO}(D, Op, Rel)$ for some set of operations Op and set of relations Rel.
- Example: Presburger arithmetic $(\mathbb{N}, \{0, 1, +\}, \{\leq\})$.
- Specifying a set of states: A formula $f(X)$
- Specifying a relation between states: A formula $R(X, X')$
- Programs are annotated with assumptions and assertions (about the set of states at particular control locations)

Checking Safety Properties

$$Init \Rightarrow \Box Safe$$

- Find and auxiliary inductive invariant Inv:

$$\begin{array}{rcl} Init & \Rightarrow & Inv \\ Inv & \Rightarrow & Safe \\ post(Inv) & \Rightarrow & Inv \end{array}$$

  or alternatively

$$Inv \Rightarrow \neg pre(\neg Inv)$$

- Reachability analysis / Synthesis of strongest inductive invariant:

$$post^*(Init) \Rightarrow Safe$$

Issues:

Representation of sets of configurations, deciding entailment, compute post/pre-images, compute reachability sets.

Models = Finite-State Automata

- Reachability is (obviously) decidable
- Model checking against temporal logics is also decidable
  - Reducible to reachability queries and cycle detection problems.
  - CTL : $|Model| \cdot |Formula|$
  - LTL : $|Model| \cdot Exp(|Formula|)$
    - Automata-based approach [Vardi, Wolper 96]
    - Associate with a formula ϕ and automaton A_ϕ s.t. $L(A_\phi) = [\![\phi]\!]$
    - Check emptiness of $L(M) \cap L(A_{\neg\phi})$

Main Problem: State-space explosion !!

- Boolean variables: $2^{\# Variables}$ states
- Concurrent systems: $2^{\# States}$ global states

Partial-Order Techniques

- Asynchrony $\Rightarrow$ a huge number of interleavings
- Several interleaving can be undistinguishable
- $\Rightarrow$ Consider only one representative of all equivalent interleavings

  Godefroid, Wolper, Valmari, Peled ... 90's

- Tools: SPIN [Holzmann, 8+,9-] ...
- An alternative approach: Petri nets (compact representation of concurrent systems)
- Solve reachability/MC queries on finite unfoldings of Petri nets

  Mc Millan, Esparza, ...

Symbolic Analysis

- Boolean variables $X = \{x_1, \ldots, x_n\}$
- Set of states = a boolean formula $f(x_1, \ldots x_n)$
- Transition relation = a boolean formula $T(x_1, \ldots x_n, x_1', \ldots x_n')$

$post^*(S)/pre^*(S)$: Compute $F_0, F_1, F_2, \ldots$ until $F_{i+1} \Rightarrow F_i$

$$\begin{array}{rcl} F_0 & = & f_S(X) \\ F_{i+1} & = & X_i \vee post/pre(F_i) \end{array}$$

Where

$$\begin{array}{rcl} post(f) & = & \exists Y.\, f(Y) \wedge T(Y, X) \\ pre(f) & = & \exists Y.\, f(Y) \wedge T(X, Y) \end{array}$$

Issue: Compact representation of boolean formulas ??

Mc Millan et al. 92 : Use Bryant's Binary Decision Diagrams.
Tool: SMV.

Efficient Representations: BDD's

- Fix an ordering between variables
- Idea: Binary decision trees + sharing + eliminating redundant tests
- Can be exponentially more concise than explicit representations
- Canonical representations
- Similar to deterministic (acyclic) finite state automata over the alphabet $\{0, 1\}$
- Efficient implementation: one single representation of each sub-dag in the memory

Many efficient BDD packages are available.

Size of the BDD's

Let $X = \{x_1, \ldots, x_n\}$. Consider the formula:

$$\bigwedge_{i=1}^{n} x_i = y_i$$

- $x_1 < y_1 < x_2 < y_2 < \ldots < x_n < y_n$.

  Linear size representation:
  Check successively $x_i = y_i$ equalities

- $x_1 < \ldots < x_n < y_1 < \ldots < y_n$.

  Exponential size representation:
  Must memorize values of all the x_i's

Bounded Model Checking

Biere, Clarke, ...

- Fix a bound K.
- Detect bugs using path of length at most K
- Encode as a boolean formula and submit to a SAT solver.
- Reachability:

$$Init(X_0) \wedge T(X_0, X_1) \wedge \cdots \wedge T(X_{k-1}, X_k) \wedge \bigvee_{i=0}^{k} BAD(X_i)$$

- Fair cycle detection:

$$Init(X_0) \wedge T(X_0, X_1) \wedge \cdots \wedge T(X_{k-1}, X_k) \wedge \bigvee_{i=0}^{k} REP(X_i) \wedge T(X_k, X_i)$$

- Performs better than BDD-based methods for bug detection.
- Completeness: $K \leq$ the longest cycle-free path in the state graph

Infinite-State Systems

- Real-time systems
- Programs with integer/real variables
- Recursive procedure calls
- Dynamic creation of threads/processes
- Arrays, dynamic data structures

Question: How to reason about infinite state spaces ?

Symbolic Reachability Analysis

- Data domain D (integers, reals, words, terms, ...)
- Variables $X = \{x_1, \ldots, x_n\}$ over D
- Set of states = a formula $f(X)$ of FO(D, Op, Rel)
- Transition relation = a formula $T(X, X')$ of FO(D, Op, Rel)

$post^*(S)/pre^*(S)$: Compute $F_0, F_1, F_2, \ldots$ until $F_{i+1} \Rightarrow F_i$

$$\begin{array}{rcl} F_0 & = & f_S(X) \\ F_{i+1} & = & X_i \vee post/pre(F_i) \end{array}$$

Where

$$\begin{array}{rcl} post(f) & = & \exists Y.\, f(Y) \wedge T(Y, X) \\ pre(f) & = & \exists Y.\, f(Y) \wedge T(X, Y) \end{array}$$

Issue: Compact representations ? Termination ??!!

Termination of Backward Analysis: Monotonic Systems

Abdulla et al., Finkel et al.

- Well-quasi ordering $\preceq$ on states: $\forall c_0, c_1, c_2, \ldots, \ \exists i < j, \ c_i \preceq c_j$
- $\Rightarrow$ Each set has a finite number of minimals
- $\Rightarrow$ Upward-closed sets are definable by their minimals
- Monotonicity: $\preceq$ is a simulation relation

$$\forall c_1, c_1', c_2. \ \big((c_1 \longrightarrow c_1' \text{ and } c_1 \preceq c_2) \Rightarrow \exists c_2'. \ c_2 \longrightarrow c_2' \text{ and } c_1' \preceq c_2'\big)$$

- $\Rightarrow$ *pre* and *pre** -images of $\preceq$-upward closed sets are $\preceq$-upward closed
- Reachability of upward-closed sets (coverability) is decidable:

  Given an UC U, the backward reachability analysis terminates:
  Collect iteratively all minimals of pre$^*(U)$

Monotonic Systems: Examples

- Vector addition systems with states (Petri nets)
  - Operations: $c := c + 1$, $c > 0/c := c - 1$
  - WQO: usual order on natural numbers
- Lossy fifo channel systems
  - Operations: send, receive to a channel + lossyness
  - WQO: substring relation
- Other examples
  - Broadcast protocol
  - Timed Petri nets
  - etc

Finite Bisimulations

- S a set of states.
- $R \subseteq S \times S$ is a bisimulation iff R is symmetrical and $(s_1, s_2) \in R$ iff

$$\forall a.\ s_1 \xrightarrow{a} s_1' \Rightarrow \exists s_2'.\ s_2 \xrightarrow{a} s_2' \text{ and } (s_1', s_2') \in R$$

- Preserves all usual properties.
- Symbolic minimal model generation (partition refinement algorithm) [B., Fernandez, Halbwachs 90]

  Ingredients: Pre-image, Intersection, Complementation

- Finite bisimulation $\Rightarrow$ Termination $\Rightarrow$ Decidability of MC
- Backward reachability analysis terminates.
- Used in many contexts, e.g., timed systems [Alur, Halbwachs, ...], hybrid systems [Henzinger et al., 9+]

Timed Automata

Alur & Dill 90

- FSA + real-valued clocks
- Dynamic:

  Time progress in control states +
  Instantaneous jumps between states

- Constraints on clocks:

  Conjunctions of $x \leq c$ *or* $x - y \leq c$, c *is an integer constant.*

- Type of constraints:

  Invariants associated with states + transition guards.

- Clocks can be reseted on transitions.

Regions

Let C be the maximal constant in the constraint of the automaton.
Let $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{R}_{\geq 0}$, the equivalence class $[\vec{x}]$ is characterized by

- Integer bounds: $(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor)$

  Partition according to the integer grid.

- Time progress: $(\mathtt{fract}(x_{i_1}), \#_1, \mathtt{fract}(x_{i_2}), \#_2, \ldots, \mathtt{fract}(x_{i_n}))$

  Add diagonals.

  where $i_1, \ldots, i_n$ is a permutation of $1, \ldots, n$, and $\#_i \in \{<, =\}$.

- Beyond C all $\lfloor x_i \rfloor$ can be abstracted to one value $(> C)$.

  Finite partition: bounded integer grid.

- Finite Region Graph: Decidable MC
- Exponential number of regions !!

Symbolic Analysis of Timed Automata: Zones and DBM's

- Let $x_1, \ldots, x_n$ be the clocks of the automaton
- Let x_0 be an additional variable always equal to 0
- Constraints:

$$\bigwedge_{i=0}^{n} x_i - x_j \#_{(i,j)} c_{(i,j)}$$

- DBM (Difference Bound Matrices):

$$M(i,j) = (\#_{(i,j)}, c_{(i,j)})$$

where $\#_{(i,j)} \in \{\leq, <\}$ and $c_{(i,j)} \in \mathbb{Z}$

- Efficient representations for symbolic computations:
  - Canonicity: Compute strongest bounds = shortest paths
  - Emptiness: existence of a negative cycle
  - Inclusion: $\leq$ — Intersection: min + can.
  - Time progress: remove upper bounds + can.
  - Reset: impose equality with x_0 + can.
- Tools: Uppaal, Kronos, ...

Acceleration

Boigelot, Wolper, B., Abdulla, Finkel, Leroux, etc.

- Let R be the transition relation of the system
- Assume that $R = R_1 \cup \cdots \cup R_n \cup R'$
- Assume we know how, given S, to compute $R_i^*(S)$, for each R_i
- Accelerated computation of reachable states:

Compute $R^*(S) = X_0 \cup X_1 \cup \cdots$ *where*

$$\begin{array}{rcl} X_0 & = & S \\ X_{i+1} & = & X_i \cup R_1^*(X_i) \cup \cdots \cup R_n^*(X_i) \cup T'(X_i) \end{array}$$

until $X_{i+1} \subseteq X_i$

  - $R_1^*, \ldots, R_n^*$ are *meta-transitions*
  - Termination is not guaranteed in general, but exact computation,
  - Can be used for under-approximate analysis.

Counter Automata

- Operations $T(X, X') : X' = AX + B$
- Is $T^n(X, X')$ representable in Presburger arithmetic ?
- No in general: $T : x' = 2x$, $T^n : x' = 2^n x$
- Conditions on A : There is a finite number of A^k, for any k.
- Example: $A = Id$, $T^n : X' = X + nB$

Abstract Analysis of Infinite-State Systems

Abstract interpretation [Cousot, Cousot, 77]

- α = abstraction function, i.e., $S \subseteq \alpha(S)$.
- Upper-approximate computation of the set of reachable states: Compute the sequence $X_0 \cup X_1 \cup \cdots$ where

$$\begin{aligned} X_0 &= S \\ X_{i+1} &= X_i \sqcup \alpha(post(X_i)) \end{aligned}$$

  until $X_{i+1} \subseteq X_i$
- Termination if no infinite increasing sequence of abstract sets
  - α *has a finite image*
  - α *is the upward closure operation wrt a WQO*
  - $\sqcup$ *is a* widening *operator (extrapolation, jumps to the limit)*

Numerical Abstract Domains

- Intervals

$$l \leq x \leq u$$

- Octagons

$$l \leq x \leq u,\ l \leq x - y \leq u,\ l \leq x + y \leq u$$

- Polyhedra

$$\sum_{i=1}^{n} a_i x_i \leq b$$

- ...

Tools: e.g., APRON [Jeannet, Miné, 09]

Non Numerical Domains

- Shape analysis [Sagiv, Reps, Willems, 96] ...

  Graphs abstracting heaps

- Shapes + Data constraints

  see for instance talk of Constantin Enea

- I will talk later about something called Abstract Regular MC

State-Space Partitioning

[Clarke, Grumberg, Long 92], [Bensalem, B., Loiseaux, Sifakis, 92]

- Let M be a infinite-state model
- Let $\sim$ be a partition of the set of states, and et $[s]$ be the $\sim$-equivalence class of s.
- $M/\!\sim$ = quotient of M w.r.t. $\sim$.

- $M/\sim$ *simulates* M:

$$\forall s.\ (M, s) \sqsubseteq (M/\!\sim, [s])$$

$\Rightarrow$ Preservation of universally path-quantified properties. (e.g., linear-time properties.)

e.g., if $\sim$ is bisimulation, then preservation of all properties.

Predicate Abstraction

Graf & Saidi 97, ...

- Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a finite set of predicates.
- Let $\sim_{\mathcal{P}}$ be the equivalence induced by $\mathcal{P}$.

$\Rightarrow$ Consider $M/{\sim_{\mathcal{P}}}$: finite abstract model.

- Constructing the abstract model:
  - A $\sim_{\mathcal{P}}$-class can be represented a boolean formula $\mathbf{b}$,
  - Given a bit vector $\mathbf{b}$, let

$$\gamma_{\mathbf{b}} = \bigwedge_{\mathbf{b}(i)=1} P_i(X) \wedge \bigwedge_{\mathbf{b}(j)=0} \neg P_j(X)$$

  - Given two formulas $\mathbf{b}$ and $\mathbf{b}'$,

$$(\mathbf{b}, \mathbf{b}') \in T/{\sim_{\mathcal{P}}} \quad \text{iff} \;\; \exists X, X'.\, \gamma_{\mathbf{b}}(X) \wedge \gamma_{\mathbf{b}'}(X') \wedge T(X, X')$$

Counter-Example Guided Abstraction Refinement

- Abstract counter-example

$$S_0 \xrightarrow{t_1} S_1 \xrightarrow{t_2} S_2 \ldots \xrightarrow{t_n} S_n \text{ with } S_n \cap BAD \neq \emptyset$$

- Compute

$$\begin{aligned} X_n &= S_n \cap BAD \\ X_k &= S_k \cap pre(X_{k+1}) \end{aligned}$$

until

- either $X_0 \neq \emptyset$: real counter-example
- or, there is $i > 0$ such that $X_i = \emptyset$: Spurious counter-example

$S_{i+1} \setminus X_{i+1}$ and X_{i+1} must be distinguished :
$\Rightarrow$ Add X_{i+1} to the set of predicates

Craig Interpolation

Let A and B be two formulas such as $A \wedge B = \texttt{false}$

An *interpolant* for (A, B) is a formula $\widehat{A}$ such that

- $A \Rightarrow \widehat{A}$
- $\widehat{A} \wedge B = \texttt{false}$
- $\widehat{A}$ refers to common variables of A and B.

Interpolants can be extracted from falsification proofs

CEGAR using Interpolation

McMillan, Jhala, ...

- Abstract counter-example

$$INIT \xrightarrow{t_1} S_1 \xrightarrow{t_2} S_2 \ldots \xrightarrow{t_n} S_n \text{ with } S_n \cap BAD \neq \emptyset$$

- Check, using an SMT solver, satisfiability of

$$f_{INIT}(X_0) \wedge t_1(X_0, X_1) \wedge t_2(X_1, X_2) \wedge \ldots t_n(X_{n-1}, X_n) \wedge f_{BAD}(X_n)$$

- If satisfiable, then real counter-example
- If not satisfiable, then for every $i \in \{1, \ldots, n\}$, consider the interpolant I_i of

$$\left(f_{INIT}(X_0) \wedge \ldots \wedge t_i(X_{i-1}, X_i),\ t_i(X_i, X_{i+1}) \wedge \ldots \wedge f_{BAD}(X_n)\right)$$

- Add all the I_i's in the set of predicates.

Procedural Programs: Recursive State Machines

- N a set of nodes. $Ent \subseteq N$ entry nodes, $Exit \subseteq N$ exit nodes.
- G a set of globals, and L a set of locals.
- Transitions:

  $n \xrightarrow{op} n'$ *where op is an operation on globals and locals, and* $n \xrightarrow{\mathtt{call}(P, en, \ell_0)} n'$

Semantics: RSM $\rightsquigarrow$ Pushdown system

- $n \xrightarrow{op} n' \rightsquigarrow \langle g, (n, \ell)\rangle \rightarrow \langle g', (n', \ell')\rangle$ where $(g', n') = op(g, n)$
- $n \xrightarrow{\mathtt{call}(P, en, \ell_0)} n' \rightsquigarrow \langle g, (n, \ell)\rangle \rightarrow \langle g, (en, \ell_0)(n', \ell)\rangle$
- $\langle g, (ex, \ell)\rangle \rightarrow \langle g, \epsilon\rangle$

Procedure Summarization

- Compute $Reach_P \subseteq (Ent \times G) \times (Exit \times G)$
- Needs the relations $R_{P,Q} \subseteq (Ent \times G \times L) \times (N \times G \times L)$
- Relations defined inductively (based on program recursive schema)
- Least fixpoint computation
- Terminates if finite-state domain. BDD-based symbolic computation.
- In general: Abstract summaries (abstract domains, widening).

Automata-based Symbolic Approach

Let P be a pushdown system

- Compute the set of backward/forward reachable configurations
- A configuration is a word $pa_1a_2\cdots a_n$, p is a control state of the PDS, and $a_1a_2\cdots a_n$ is the stack content.
- Use a finite-state automaton A_C to represent a regular set of configurations C.
- For every regular set of configurations C, $post^*(C)$ and $pre^*(C)$ are regular and effectively constructible.

Computing pre^*-image [B. Esparza, Maler 97]

- A_C has a state s_p for each control state p of P
- Compute a sequence of automata $A_0 = A_C$, A_1, ...
- $\langle p_1, a\rangle \rightarrow \langle p_2, w\rangle$ a transition of P, if $s_{p_1} \xrightarrow{w} q$ in A_i, then add $s_{p_2} \xrightarrow{a} q$ to it.
- Termination: fixed number of states.

Regular Model Checking

Abdulla, B., Jonsson, Pnueli, Saksena, Touili, Hebermehl, Vojnar, ...

- A configuration encoded as a word/tree
- Set of configurations $\rightsquigarrow$ finite-state automaton
- An action is encoded as finite-state transducer (I/O automaton)
- Reachability problem $\rightsquigarrow$

  Given automata A and B, and a transducer T, check if

$$T^*(A) \cap B = \emptyset$$

- Application to:
  - Networks of processes: Configuration = sequence/tree of local states
  - Counter automata: Encode integers as finite words over $\{0, 1\}$
  - Dynamic linked structures: Encode heaps as word / trees

RMC based verification

Generic techniques for computing (exactly/approximatively) $T^*(A)$ (or T^*)

- Acceleration techniques for some classes of regular relations
- Exact abstractions on transducers for transitive closure computations

  Abdulla, Nilsson, Jonsson, Saksena ... 0+

- Abstractions on automata with counter-example guided refinement

  B., Habermehl, Rogalewich, Vojnar 06

  - Define equivalence relations on state of automata
  - Example: Accept the same words of length $\leq k$.
  - Predicate abstraction + CEGAR: A predicate = automaton
  - Applied to the analysis of complex heap-manipulating programs

  Heaps $\rightsquigarrow$ *Tree + navigation expressions*

Challenges

- Complex theories: structures + data constraints
- Composition in procedure decisions (split according to various domains, and combine)
- Abstraction in procedure decisions (soundness + scalability)
- Complex behavior (e.g., concurrency)
- Probabilistic verification (how likely the model is correct)
- Quantitative verification (measure the quality of the implementation)
- Synthesis (program repair)